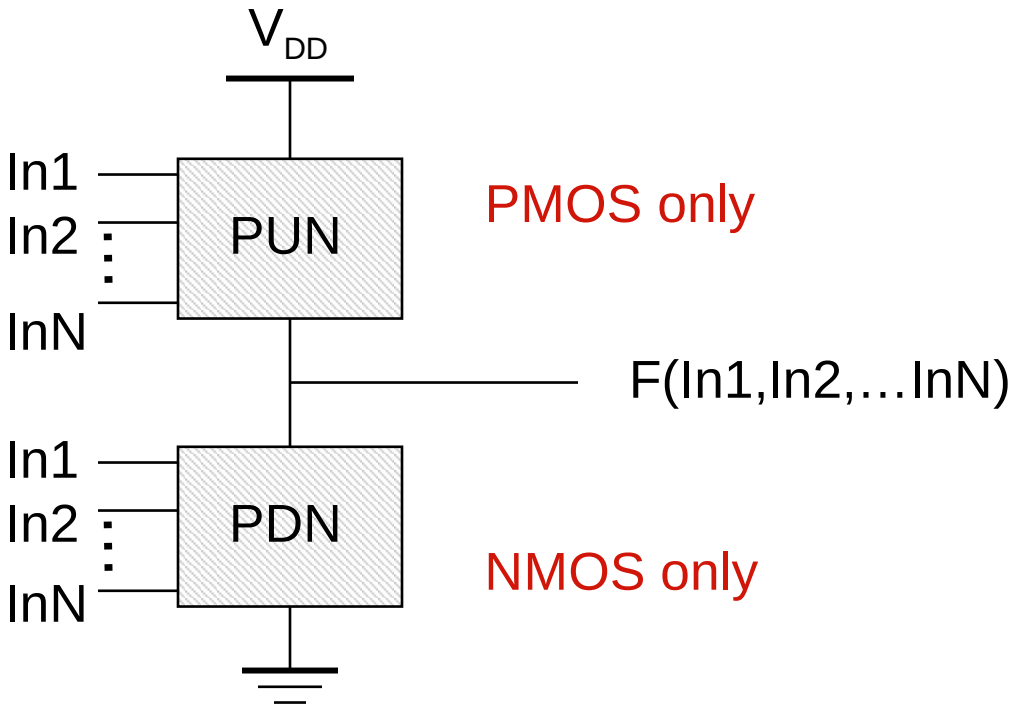
Designing Static CMOS Logic Circuits

Static CMOS Circuits

At every point in time (except during the switching transients) each gate output is connected to either V_{DD} or V_{ss} via a low-resistive path.

The outputs of the gates assume at all times the value of the Boolean function, implemented by the circuit (ignoring, once again, the transient effects during switching periods).

Static Complementary CMOS

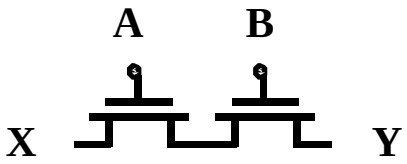


PUN and PDN are logically dual logic networks

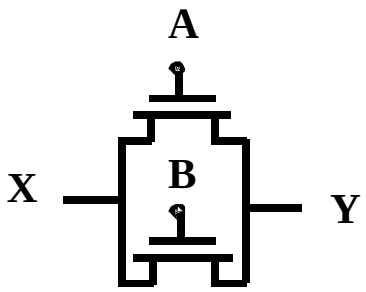
NMOS Transistors in Series/Parallel Connection

Transistors can be thought as a switch controlled by its gate signal

NMOS switch closes when switch control input is high



Y = X if A and B

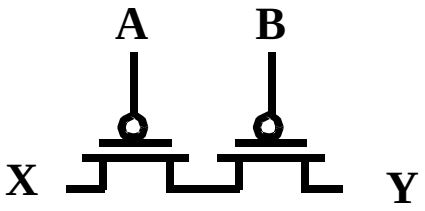


Y = X if A OR B

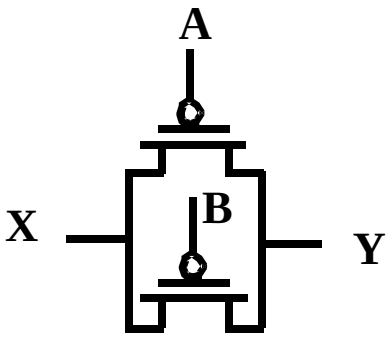
NMOS Transistors pass a “strong” 0 but a “weak” 1

PMOS Transistors in Series/Parallel Connection

PMOS switch closes when switch control input is low



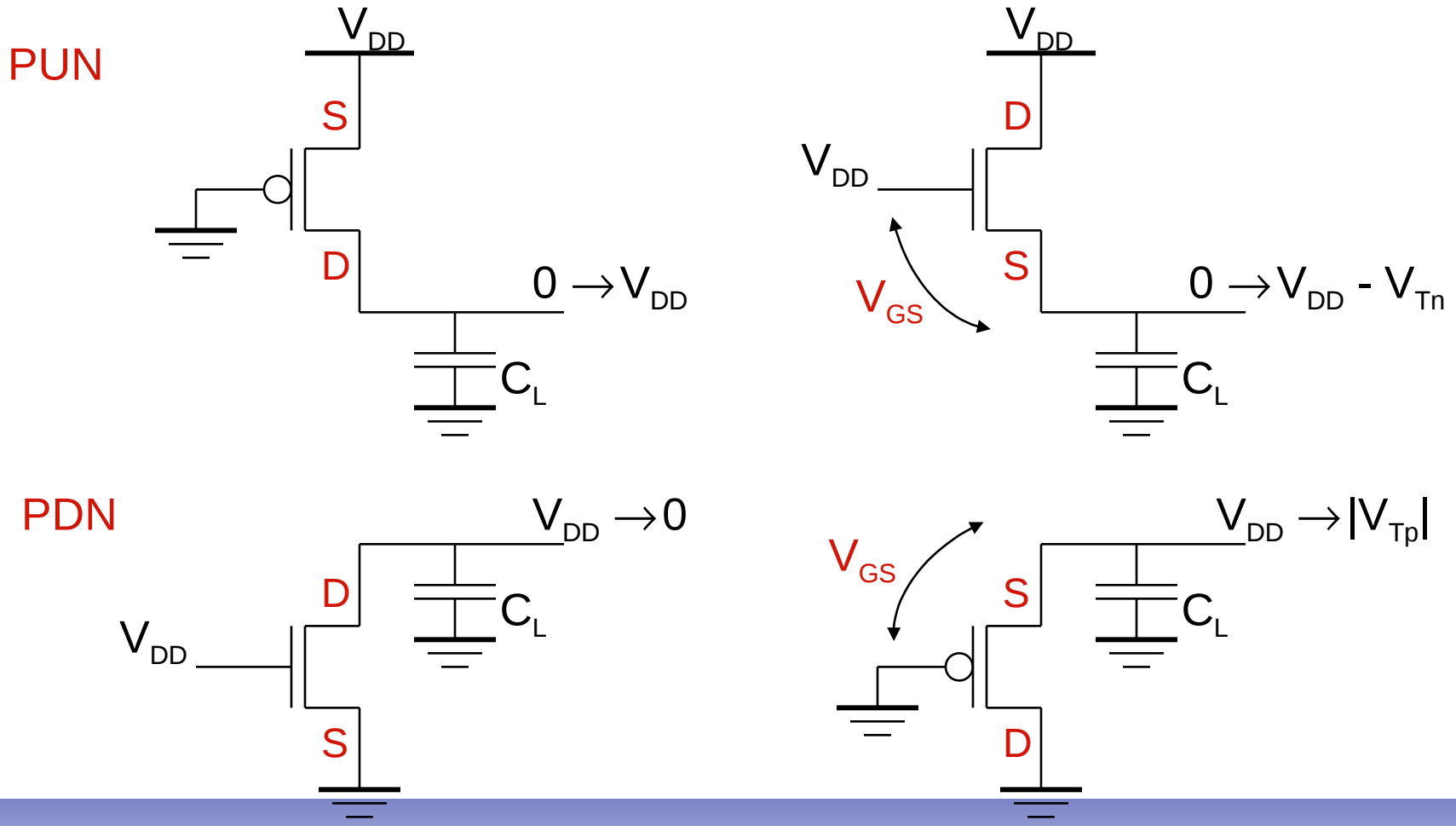
$Y = X$ if $\overline{A}$ AND $\overline{B} = \overline{A + B}$



$Y = X$ if $\overline{A}$ OR $\overline{B} = \overline{AB}$

PMOS Transistors pass a "strong" 1 but a "weak" 0

Threshold Drops



Complementary CMOS Logic Style

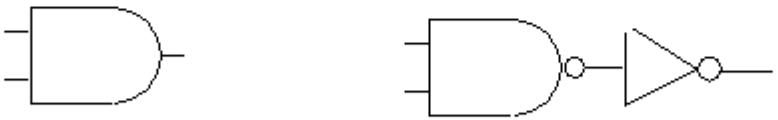
- **PUP is the DUAL of PDN**

  (can be shown using DeMorgan's Theorem's)

$$\overline{A + B} = \bar{A}\bar{B}$$

$$\overline{AB} = \bar{A} + \bar{B}$$

- **The complementary gate is inverting**

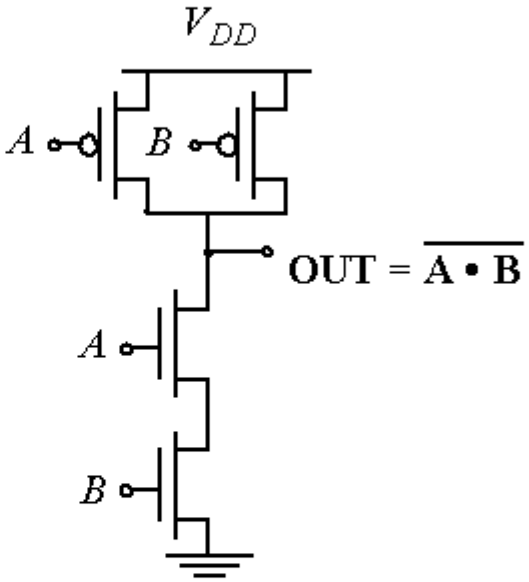


AND = NAND + INV

Example Gate: NAND

A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

Truth Table of a 2 input NAND gate



PDN: G = A B $\Rightarrow$ Conduction to GND

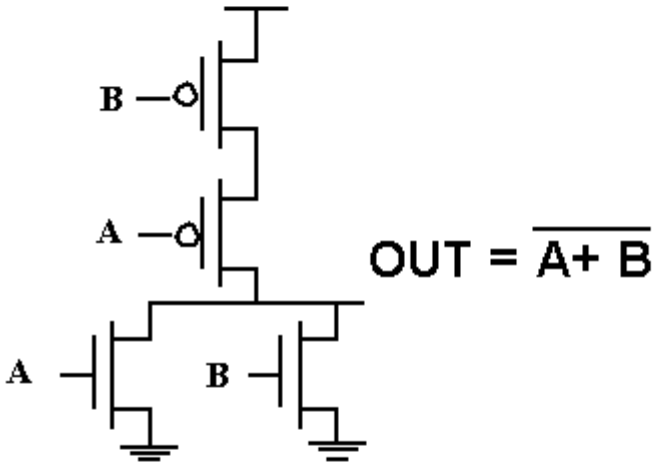
PUN: F= $\overline{A} + \overline{B} = \overline{AB} \Rightarrow$ Conduction to V_{DD}

$$\overline{G(In_1, In_2, In_3, \ldots)} \equiv F(\overline{In_1}, \overline{In_2}, \overline{In_3}, \ldots)$$

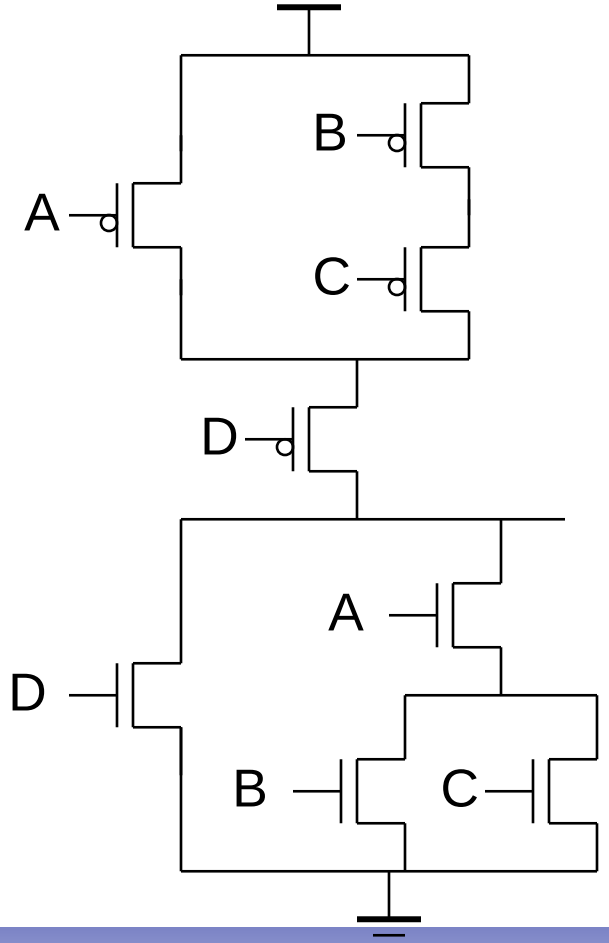
Example Gate: NOR

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

Truth Table of a 2 input NOR gate



Complex CMOS Gate



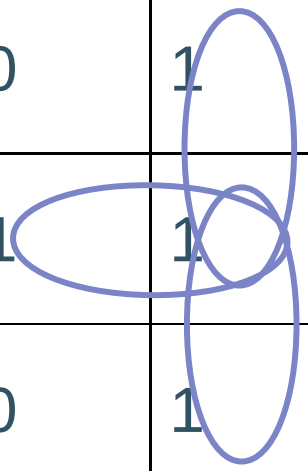
$$OUT = \overline{D + A \bullet (B + C)}$$

Constructing a Complex Gate

- Logic Dual need ***not*** be Series/Parallel Dual
- In general, many logical dual exist, need to choose one with best characteristics
- Use Karnaugh-Map to find good duals
  - Goal: find 0-cover and 1-cover with best parasitic or layout properties
  - Maximize connections to power/ground
  - Place critical transistors closest to output node

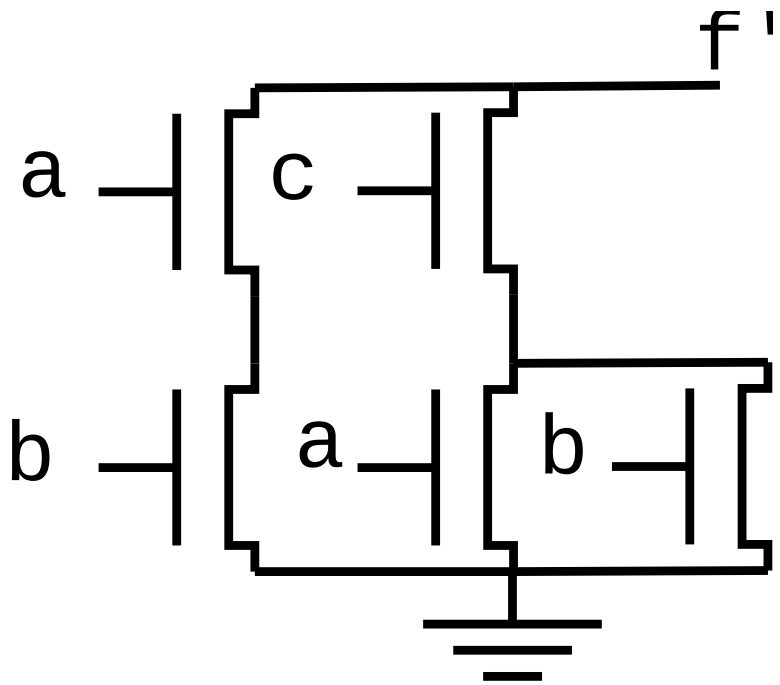
Example: Carry Gate

	C	C’
AB	0	0
AB’	0	1
A’B’	1	1
A’B	0	1



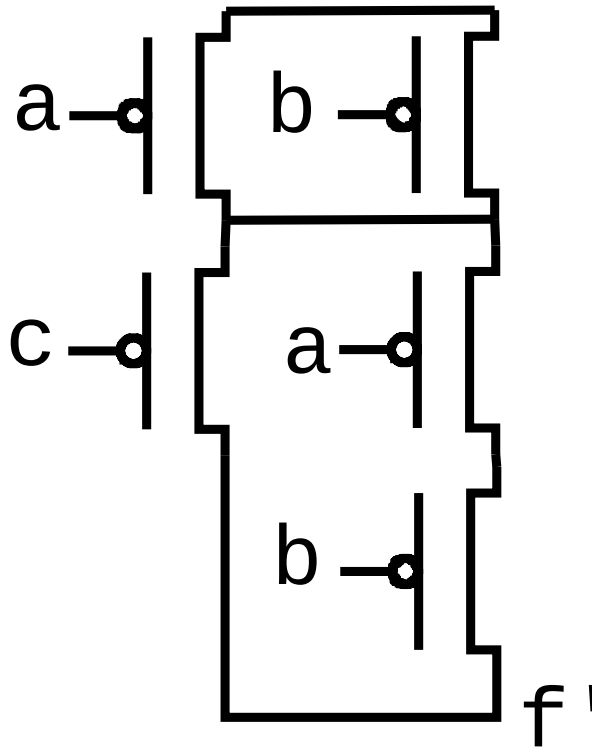
- F = (ab+bc+ac)’
- Carry ‘c’ is critical
- Factor c out:
- F=(ab+c(a+b))’
- 0-cover is n-pd
- 1-cover is p-pu

Example: Carry Gate (2)



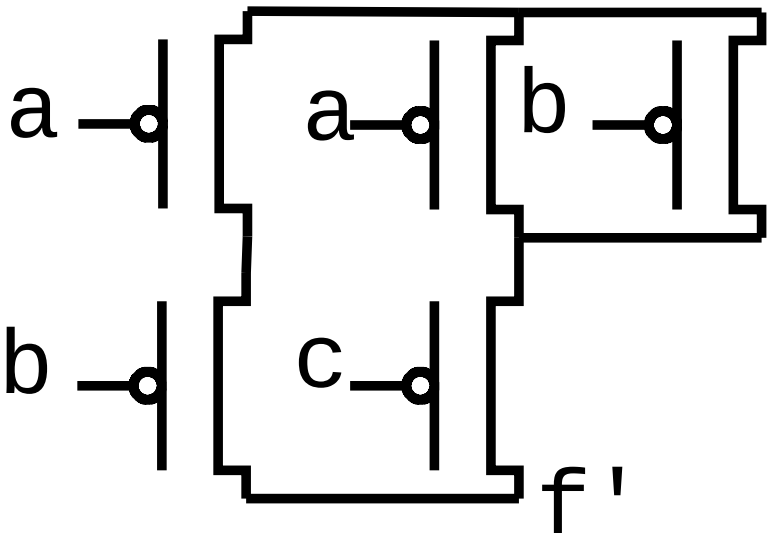
- Pull Down is easy
- Order by maximizing connections to ground and critical transistors
- For pull up – Might guess series dual– would guess wrong

Example: Carry Gate (3)



- Series/Parallel Dual
- 3-series transistors
- 2 connections to Vdd
- 7 floating capacitors

Example: Carry Gate (4)

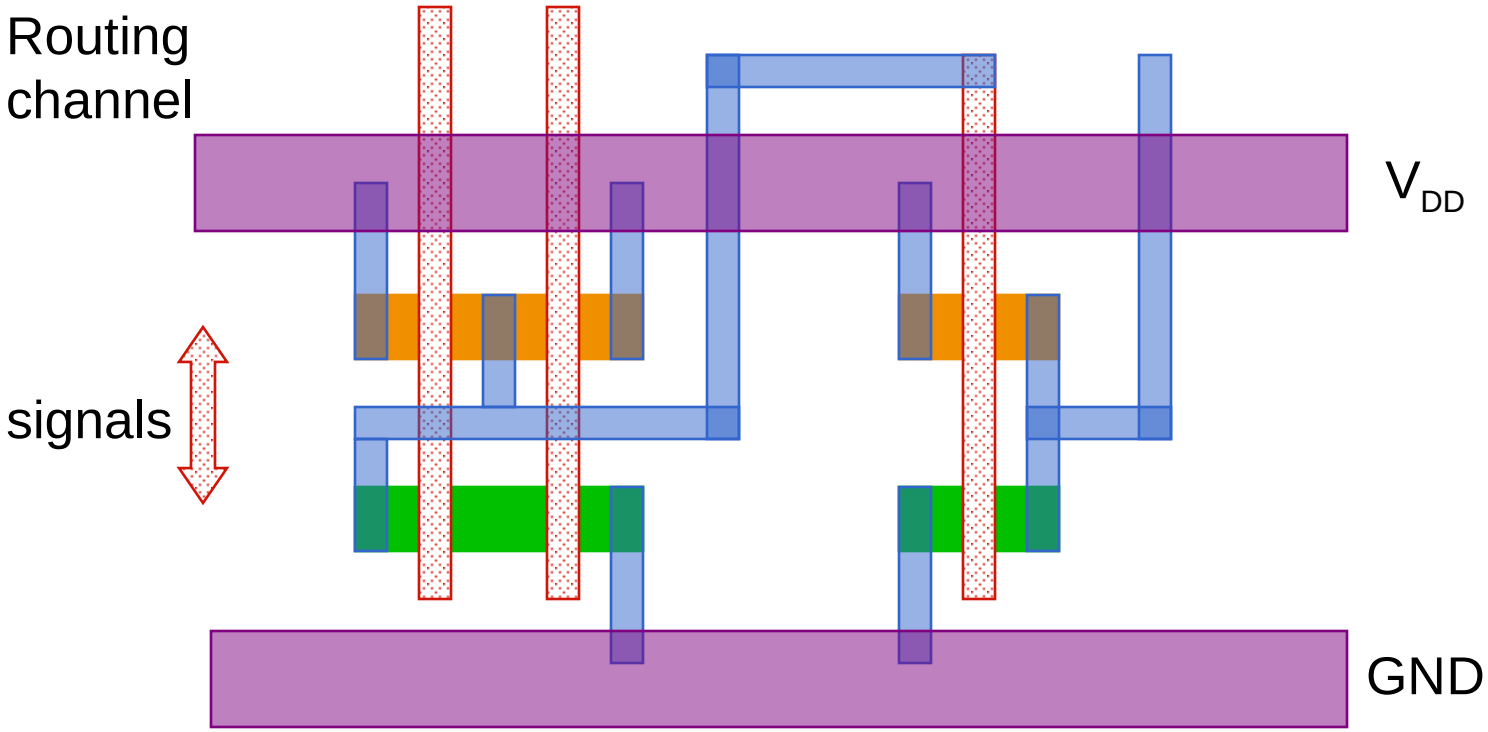


- Pull Up from 1 cover of Kmap
  - Get a'b'+a'c'+b'c'
  - Factor c' out
- 3 connections to Vdd
- 2 series transistors
- Co-Euler path layout
- Moral: Use Kmap!

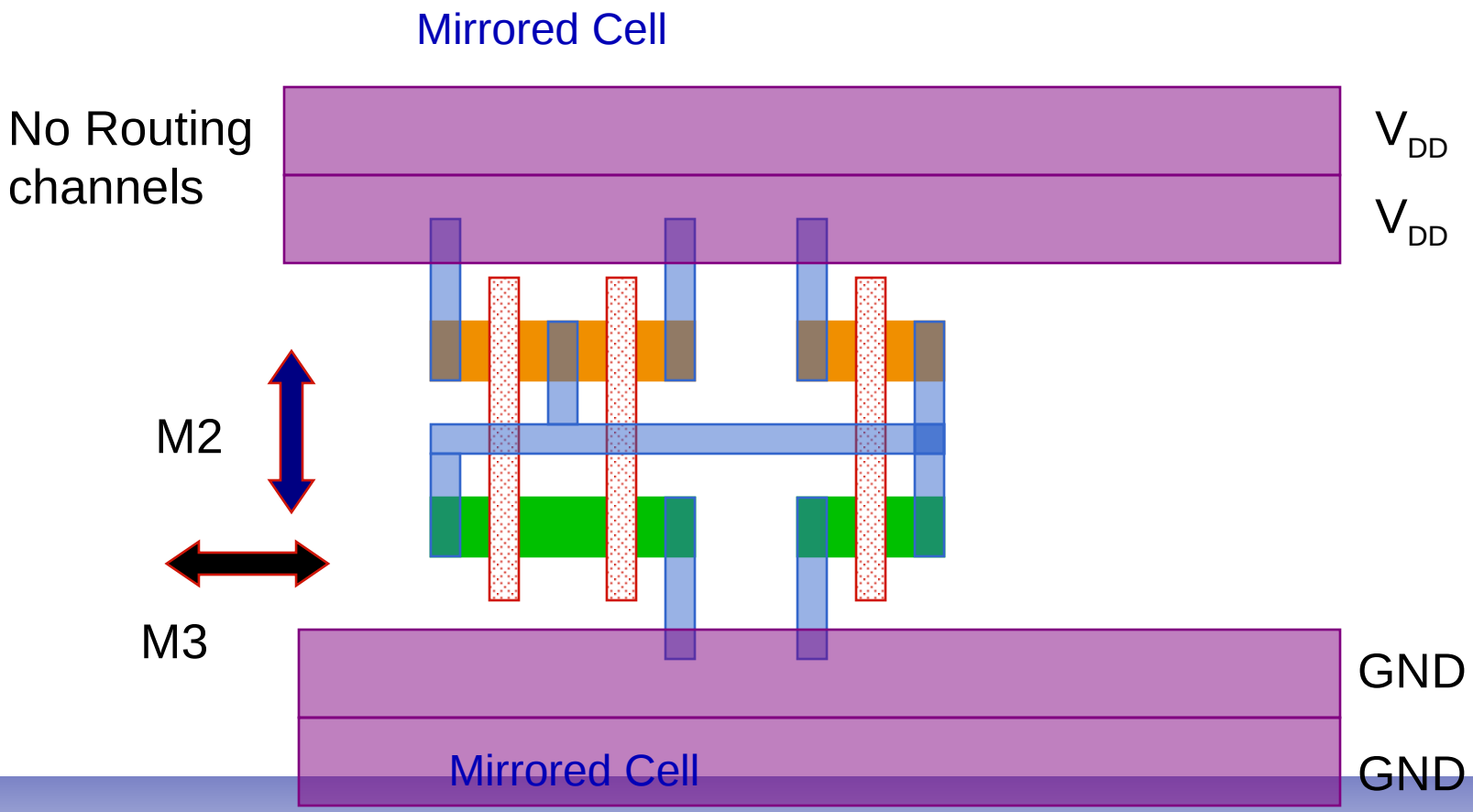
Cell Design

- Standard Cells
  - General purpose logic
  - Can be synthesized
  - Same height, varying width
- Datapath Cells
  - For regular, structured designs (arithmetic)
  - Includes some wiring in the cell
  - Fixed height and width

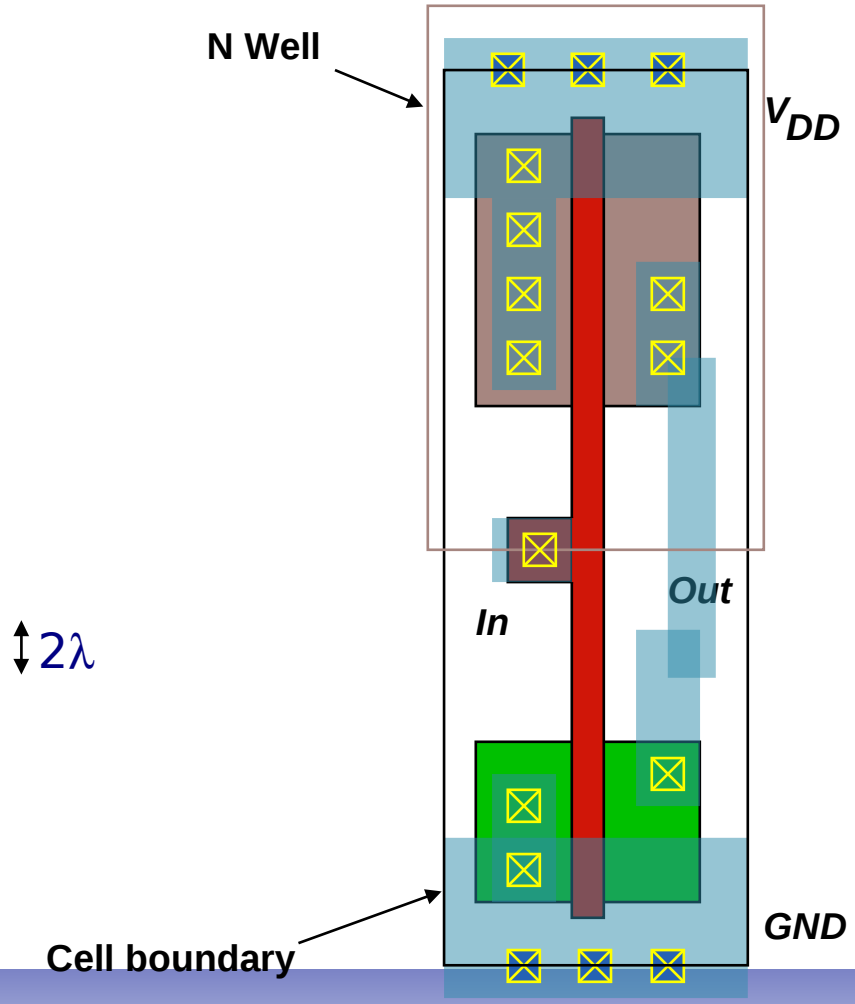
Standard Cell Layout Methodology – 1980s



Standard Cell Layout Methodology – 1990s



Standard Cells



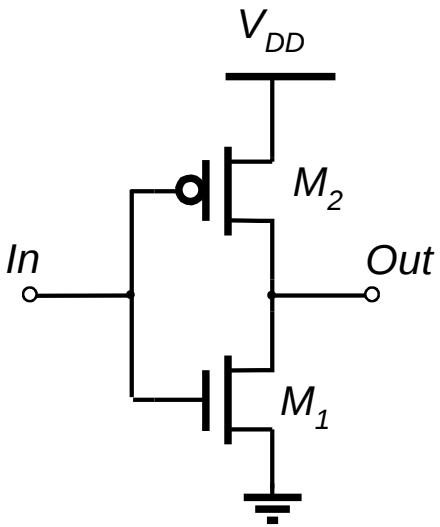
Cell height 12 metal tracks
Metal track is approx. 3λ + 3λ
Pitch =
repetitive distance between objects

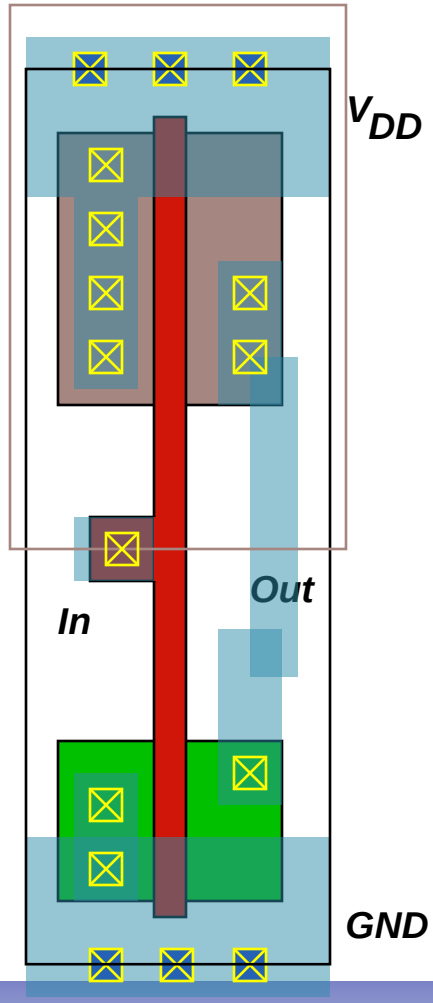
Cell height is “12 pitch”

Rails ~10λ

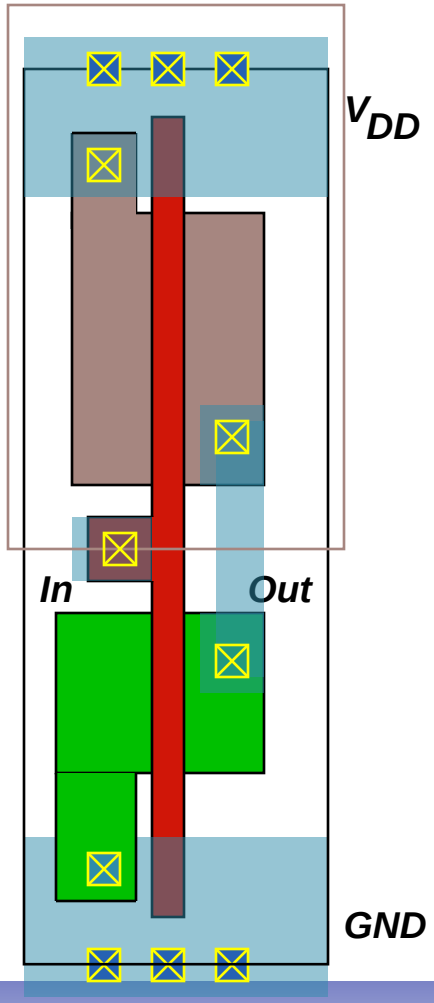
Standard Cells

With minimal diffusion routing

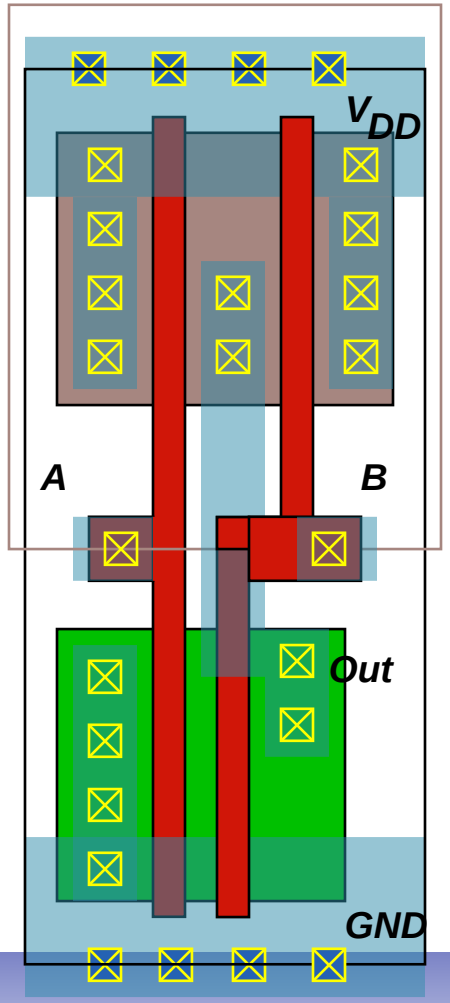




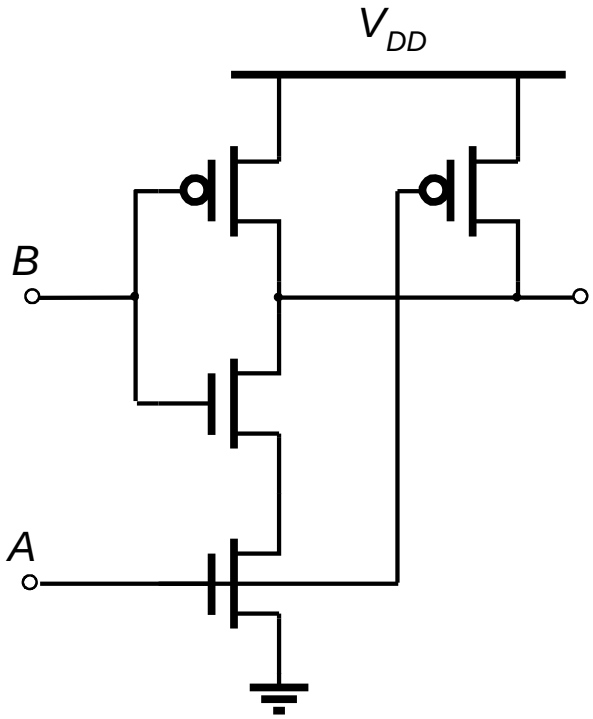
With silicided diffusion



Standard Cells



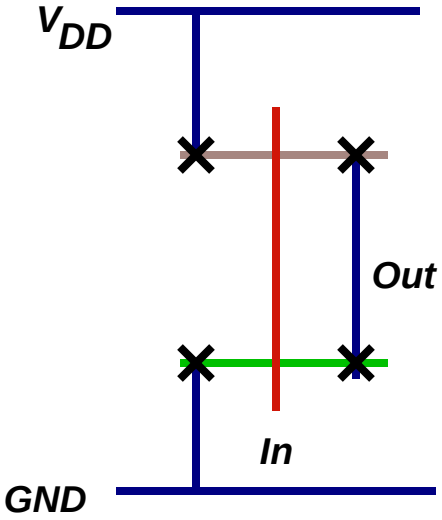
2-input NAND gate



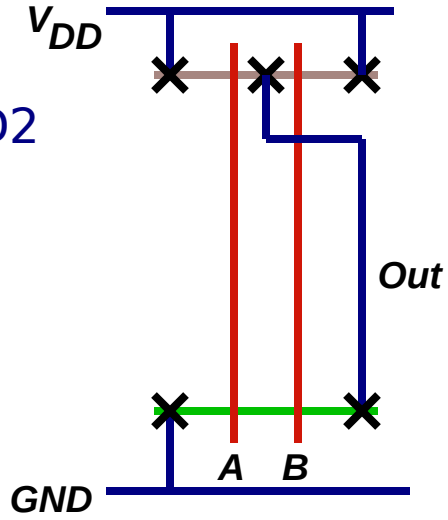
Stick Diagrams

Contains no dimensions
Represents relative positions of transistors

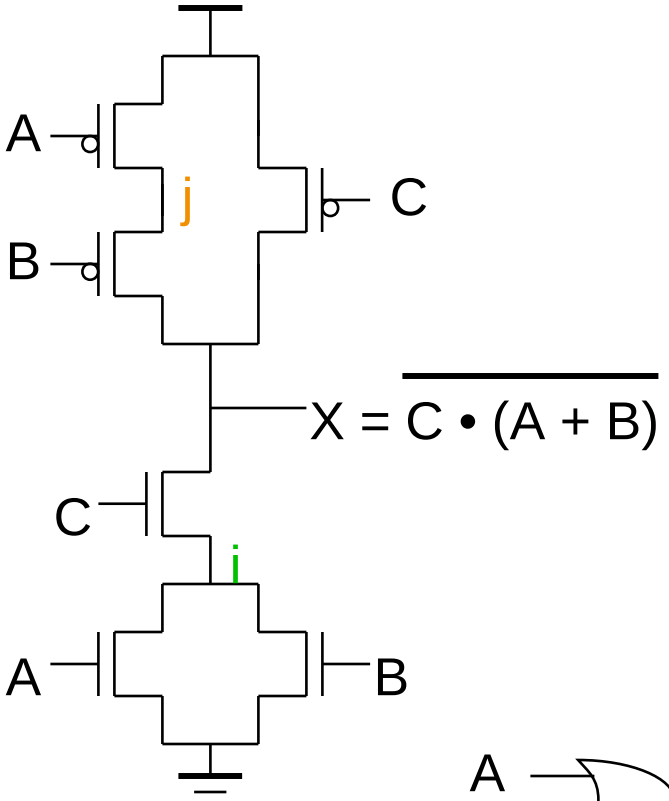
Inverter



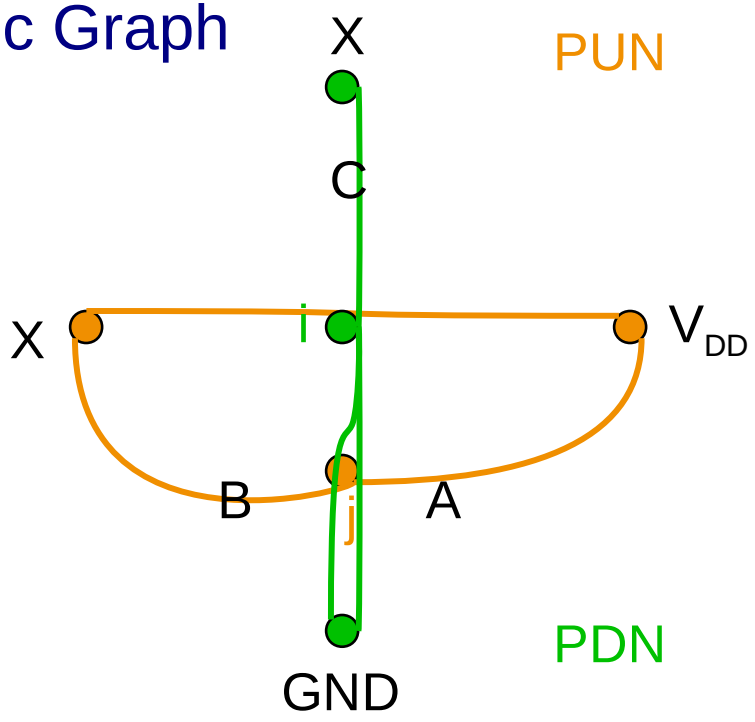
NAND2

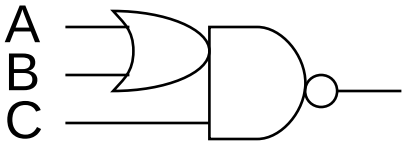


Stick Diagrams

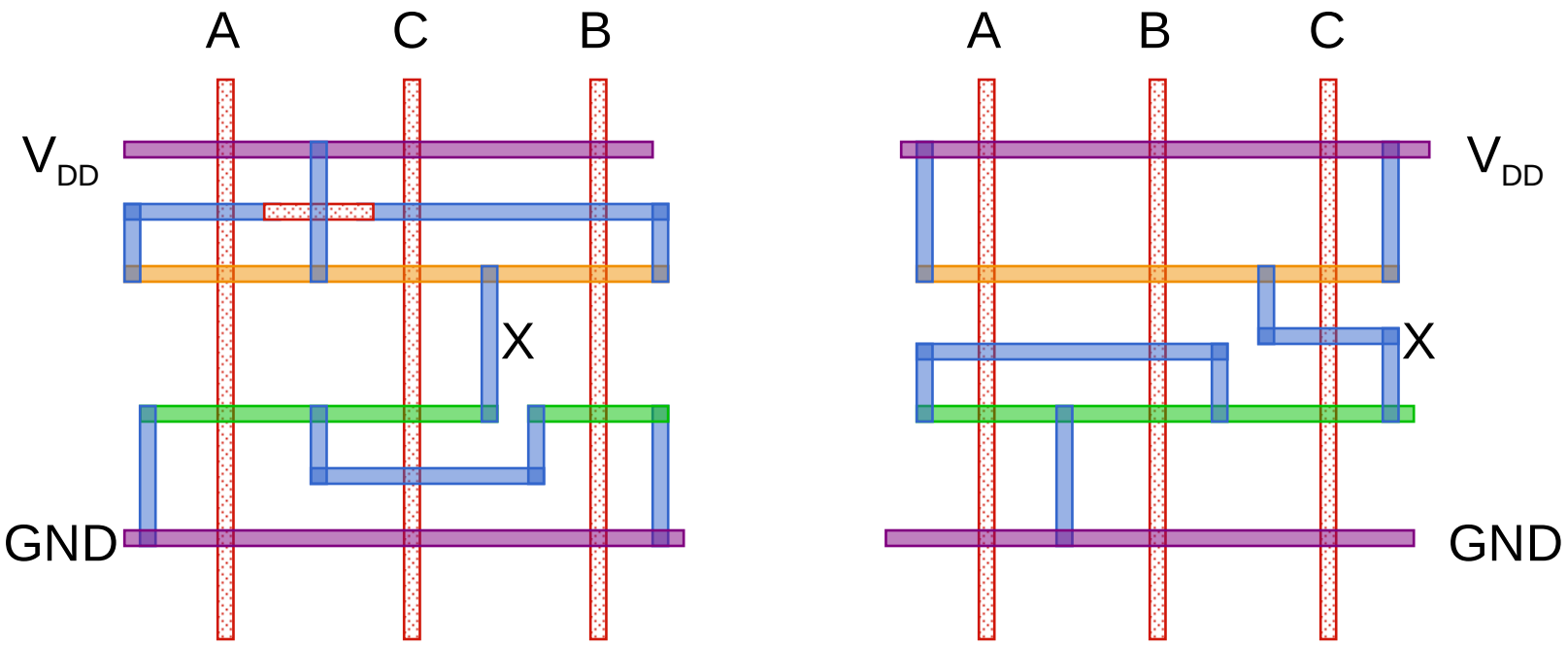


Logic Graph

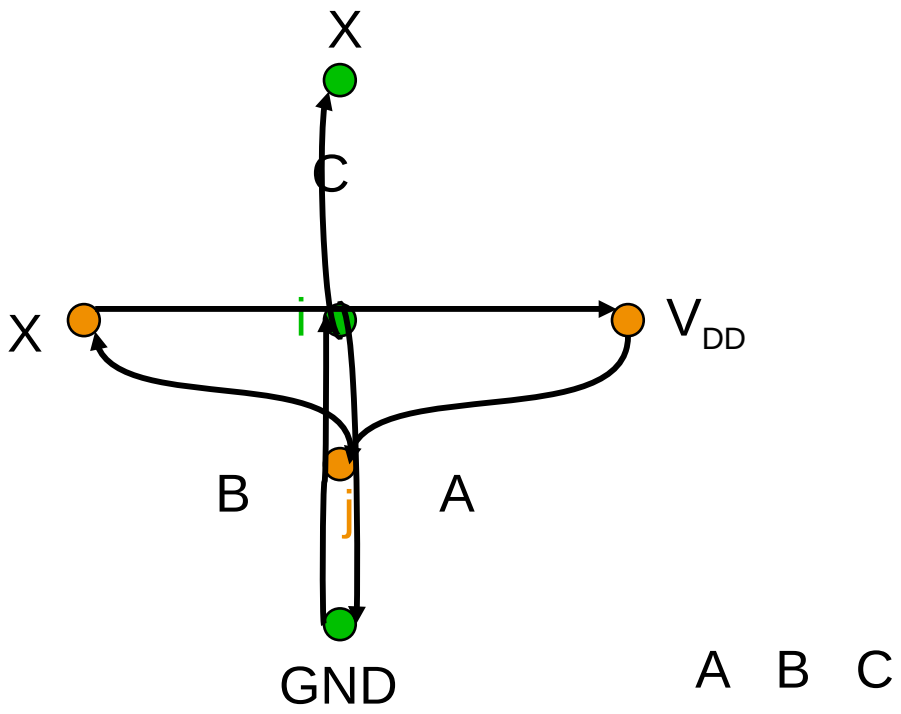




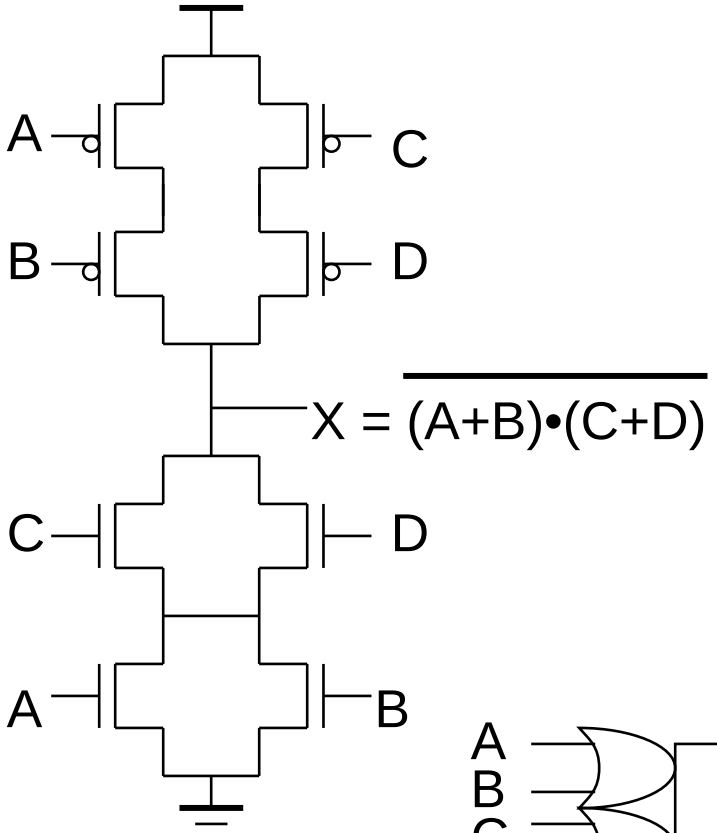
Two Versions of $\overline{C \cdot (A + B)}$



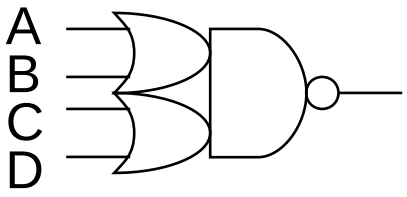
Consistent Euler Path

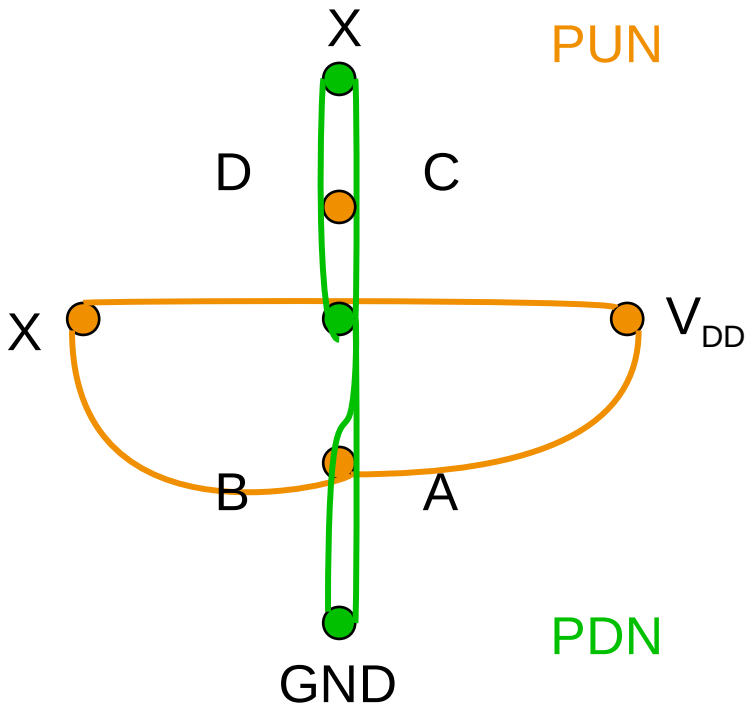


OAI22 Logic Graph

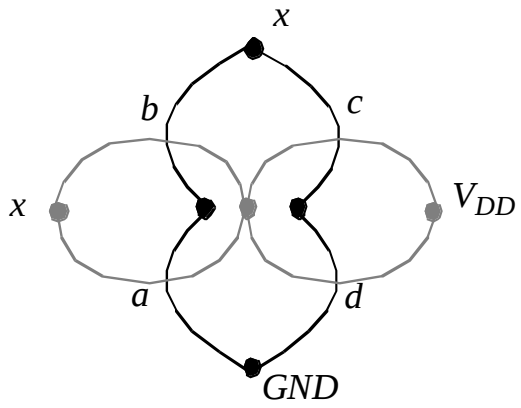


$$X = \overline{(A+B)\bullet(C+D)}$$

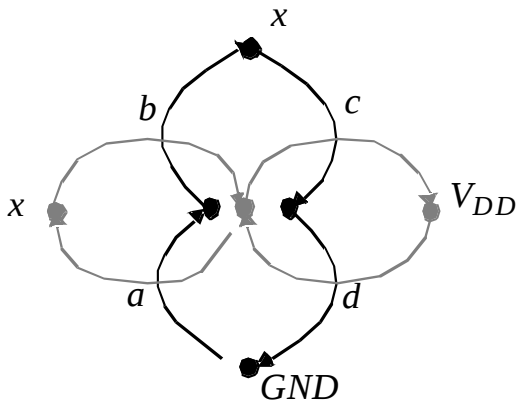




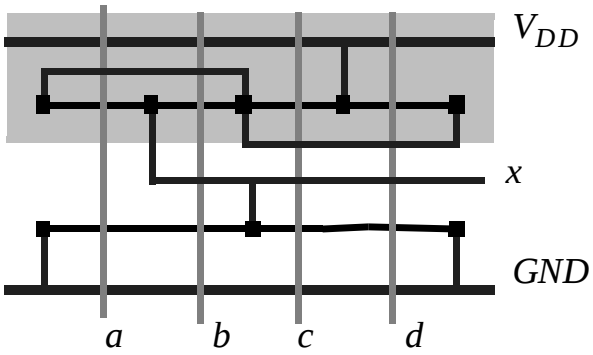
Example: x = ab+cd



(a) Logic graphs for $\overline{(ab+cd)}$



(b) Euler Paths $\{a\ b\ c\ d\}$



(c) stick diagram for ordering {a *b c d*}

Properties of Complementary CMOS Gates Snapshot

***High noise margins*:**

V_{OH} and V_{OL} are at V_{DD} and *GND*, respectively.

***No static power consumption*:**

There never exists a direct path between V_{DD} and V_{SS} (*GND*) in steady-state mode.

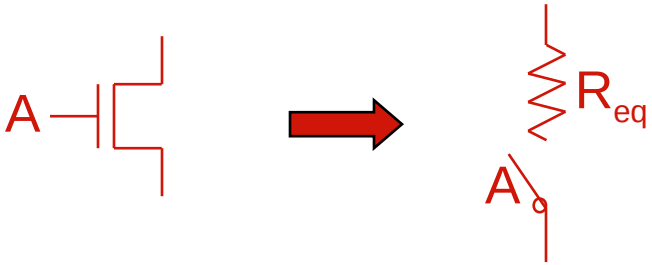
***Comparable rise and fall times*:**

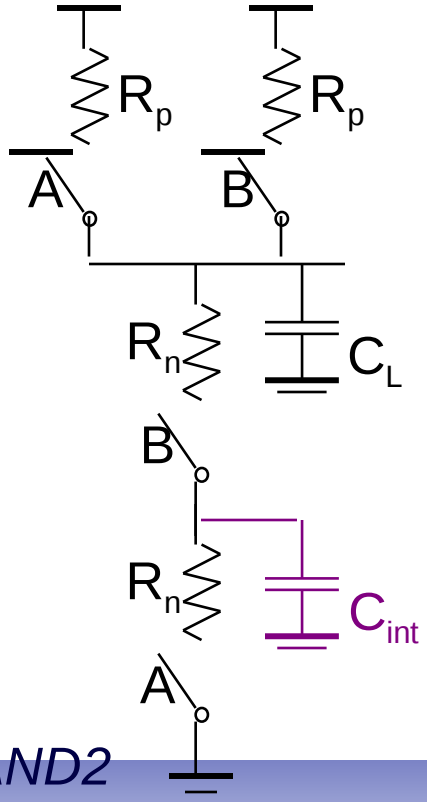
(under appropriate sizing conditions)

CMOS Properties

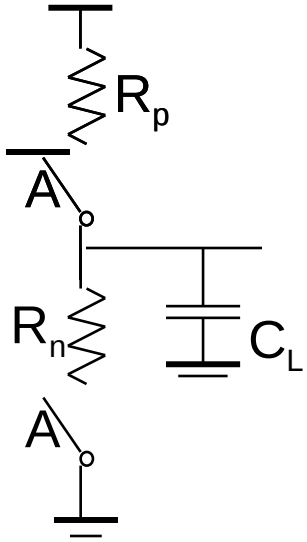
- Full rail-to-rail swing; high noise margins
- Logic levels not dependent upon the relative device sizes; ratioless
- Always a path to Vdd or Gnd in steady state; low output impedance
- Extremely high input resistance; nearly zero steady-state input current
- No direct path steady state between power and ground; no static power dissipation
- Propagation delay function of load capacitance and resistance of transistors

Switch Delay Model

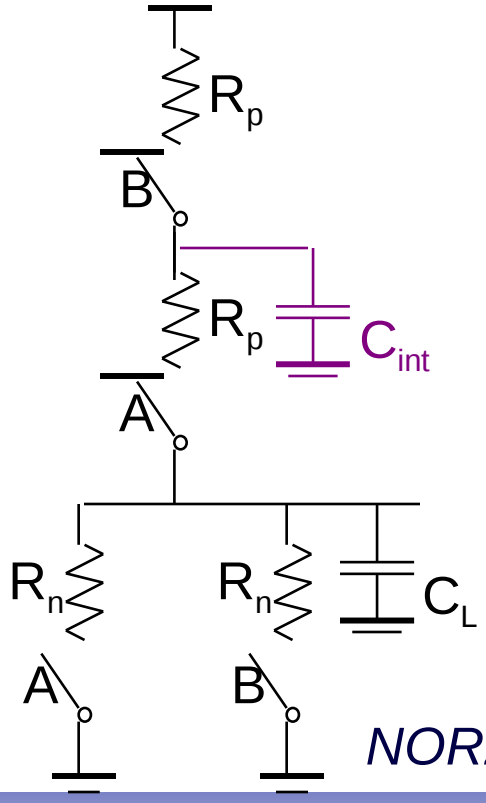




NAND2

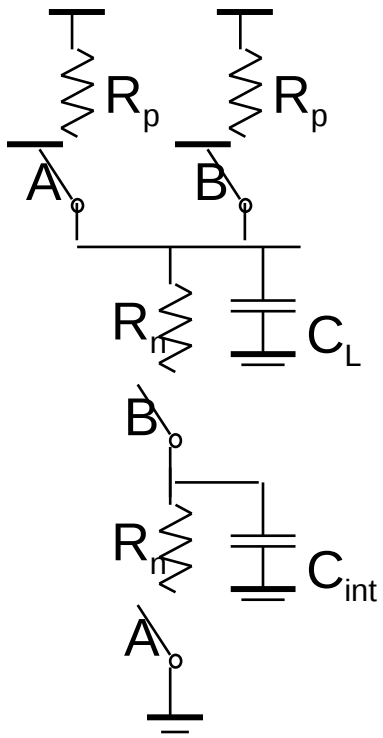


INV



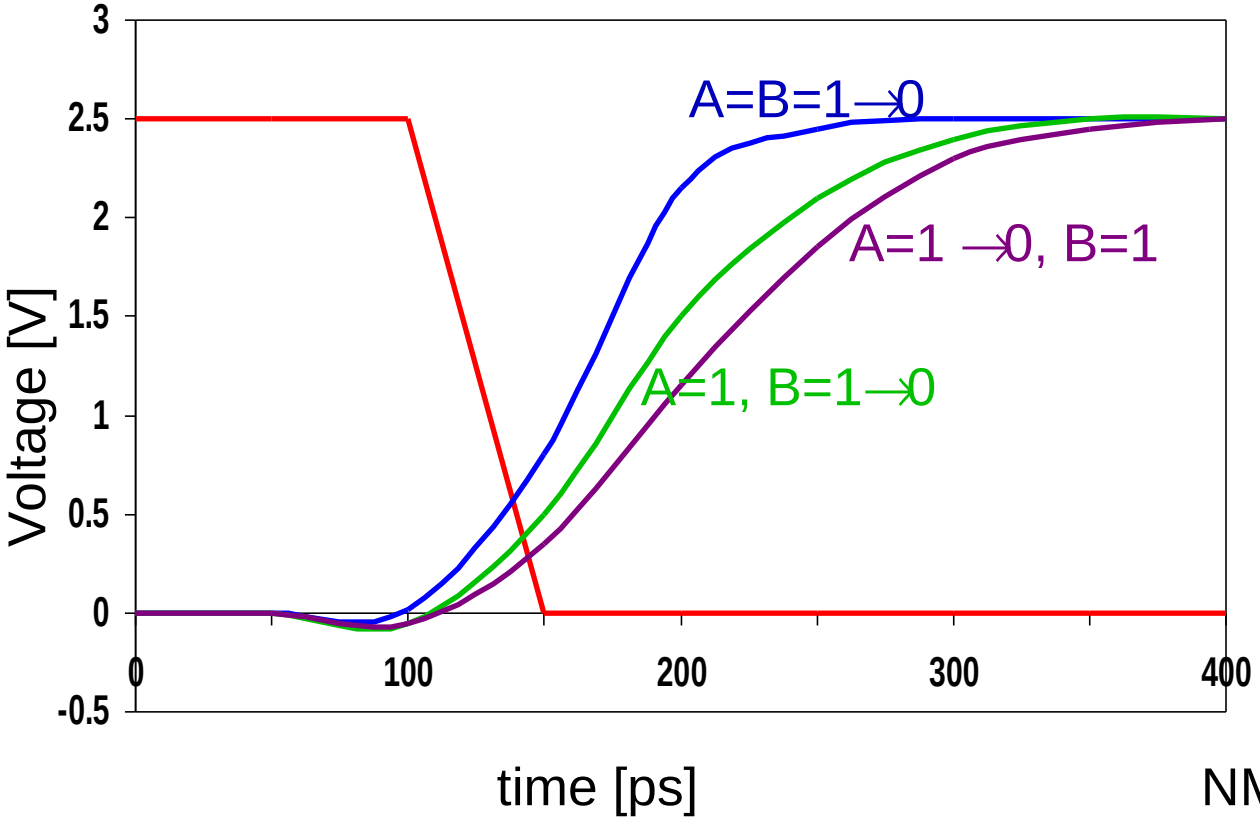
NOR2

Input Pattern Effects on Delay



- Delay is dependent on the pattern of inputs
- Low to high transition
  - both inputs go low
    - delay is 0.69 $R_p/2$ C_L
  - one input goes low
    - delay is 0.69 R_p C_L
- High to low transition
  - both inputs go high
    - delay is 0.69 $2R_n$ C_L

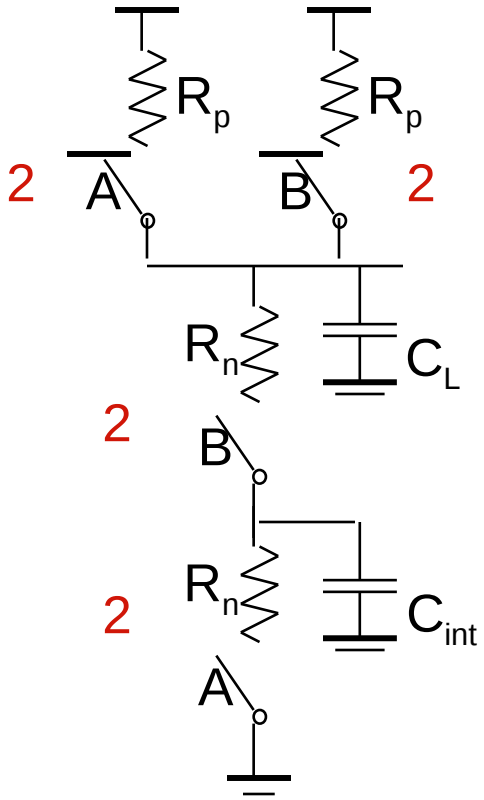
Delay Dependence on Input Patterns

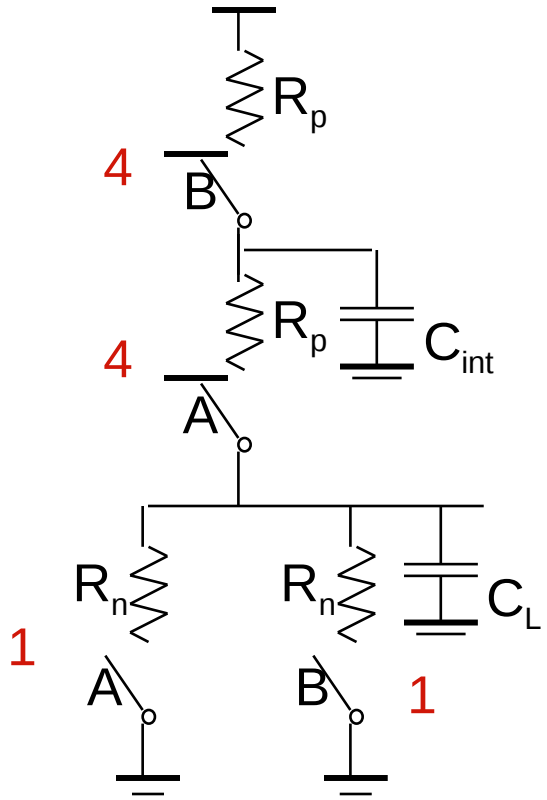


Input Data Pattern	Delay (psec)
A=B=0→1	67
A=1, B=0→1	64
A= 0→1, B=1	61
A=B=1→0	45
A=1, B=1→0	80
A= 1→0, B=1	81

NMOS = 0.5µm/0.25 µm
PMOS = 0.75µm/0.25 µm
C_L = 100 fF

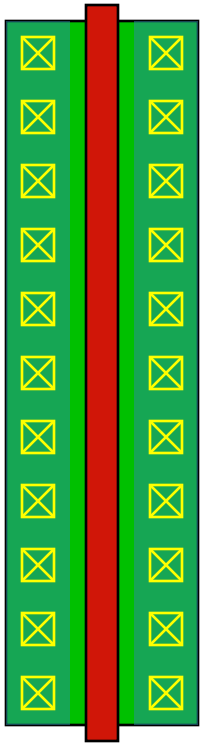
Transistor Sizing



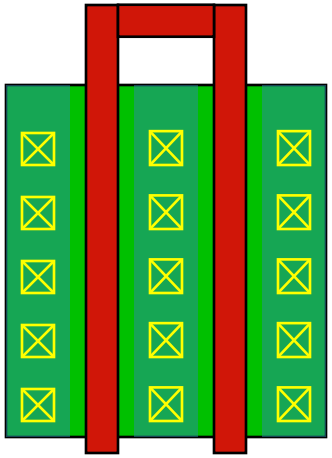


Multi-Fingered Transistors

One finger

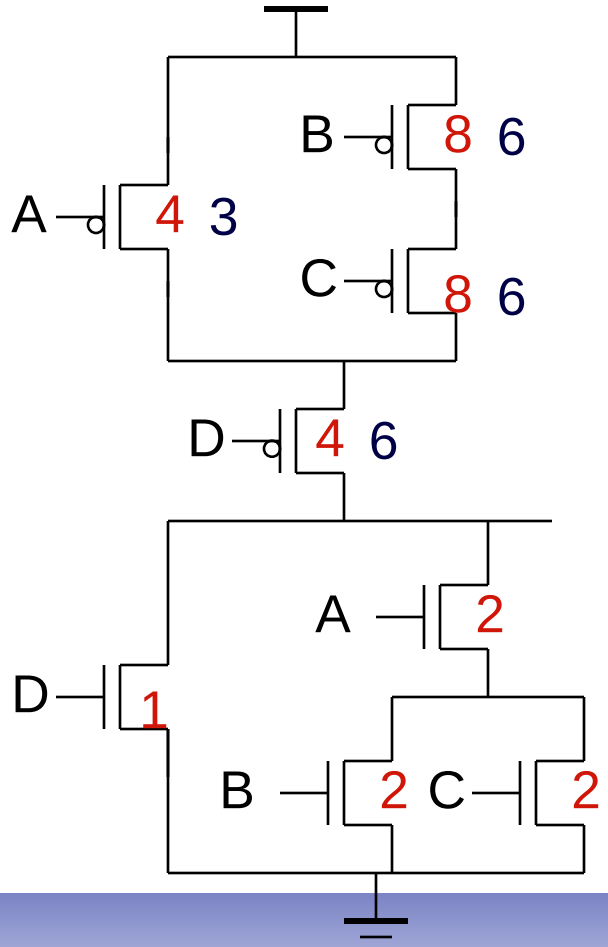


Two fingers (folded)



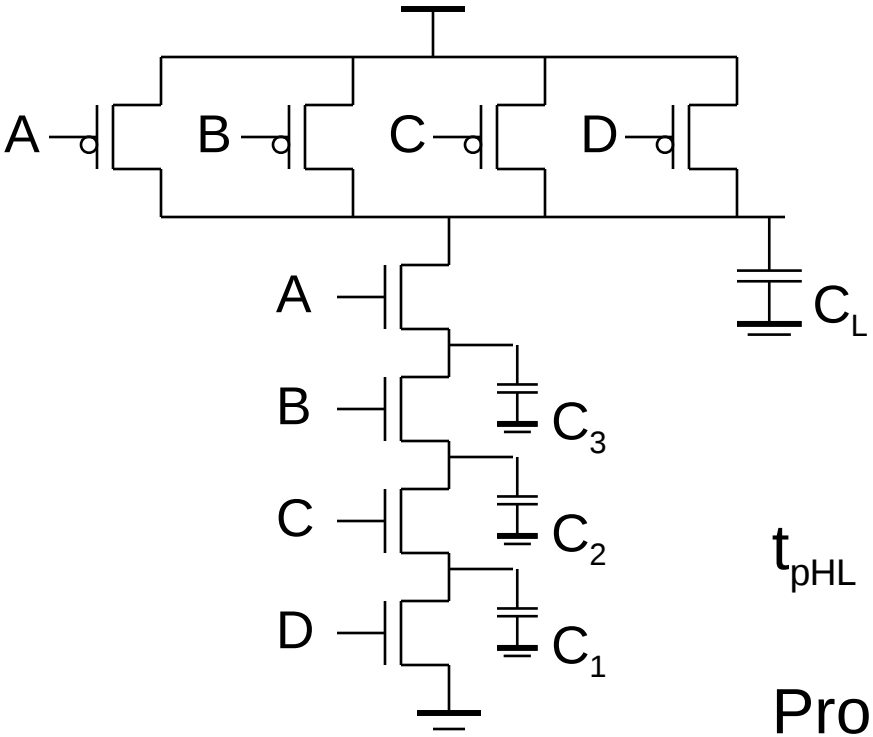
Less diffusion capacitance

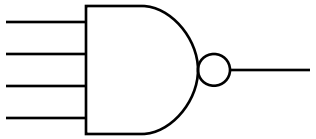
Transistor Sizing a Complex CMOS Gate



$$OUT = \overline{D + A \bullet (B + C)}$$

Fan-In Considerations



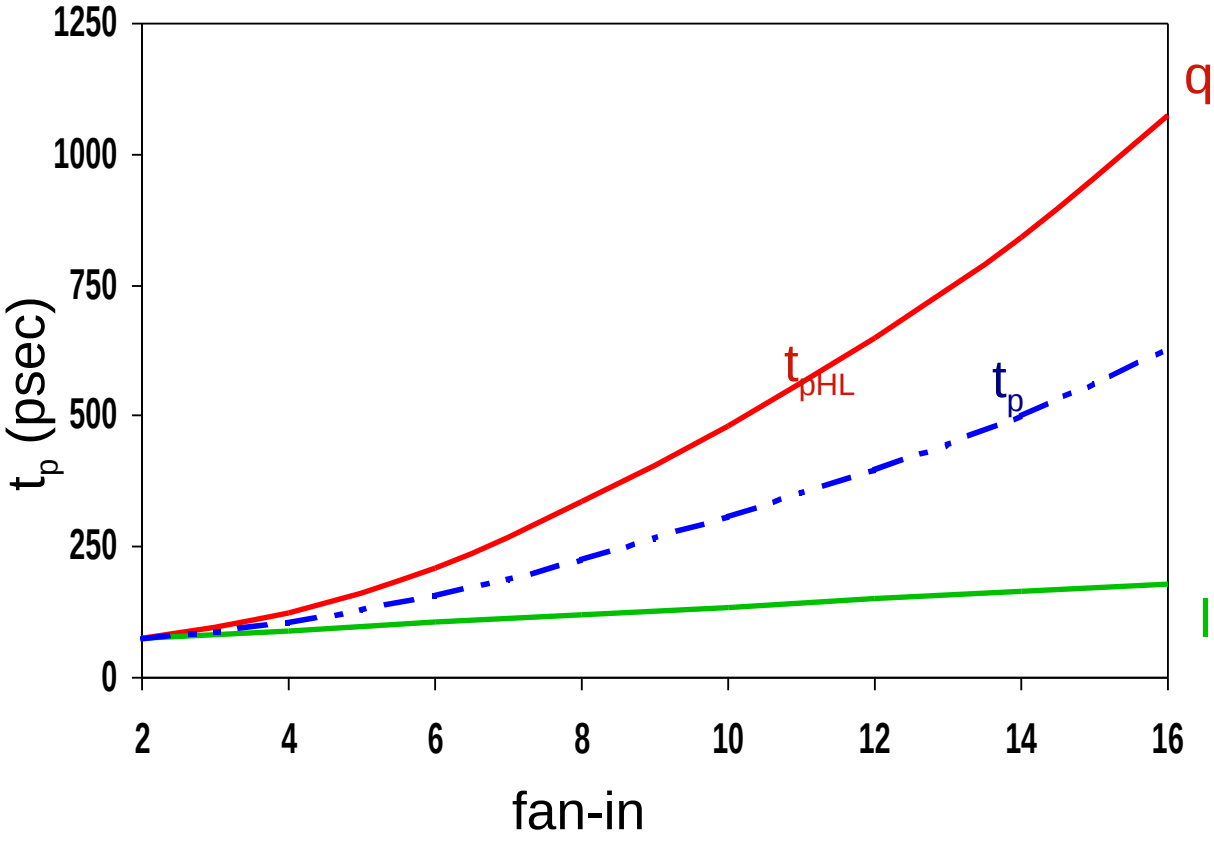


Distributed RC model
(Elmore delay)

$$t_{pHL} = 0.69\, R_{eqn}(C_1+2C_2+3C_3+4C_L)$$

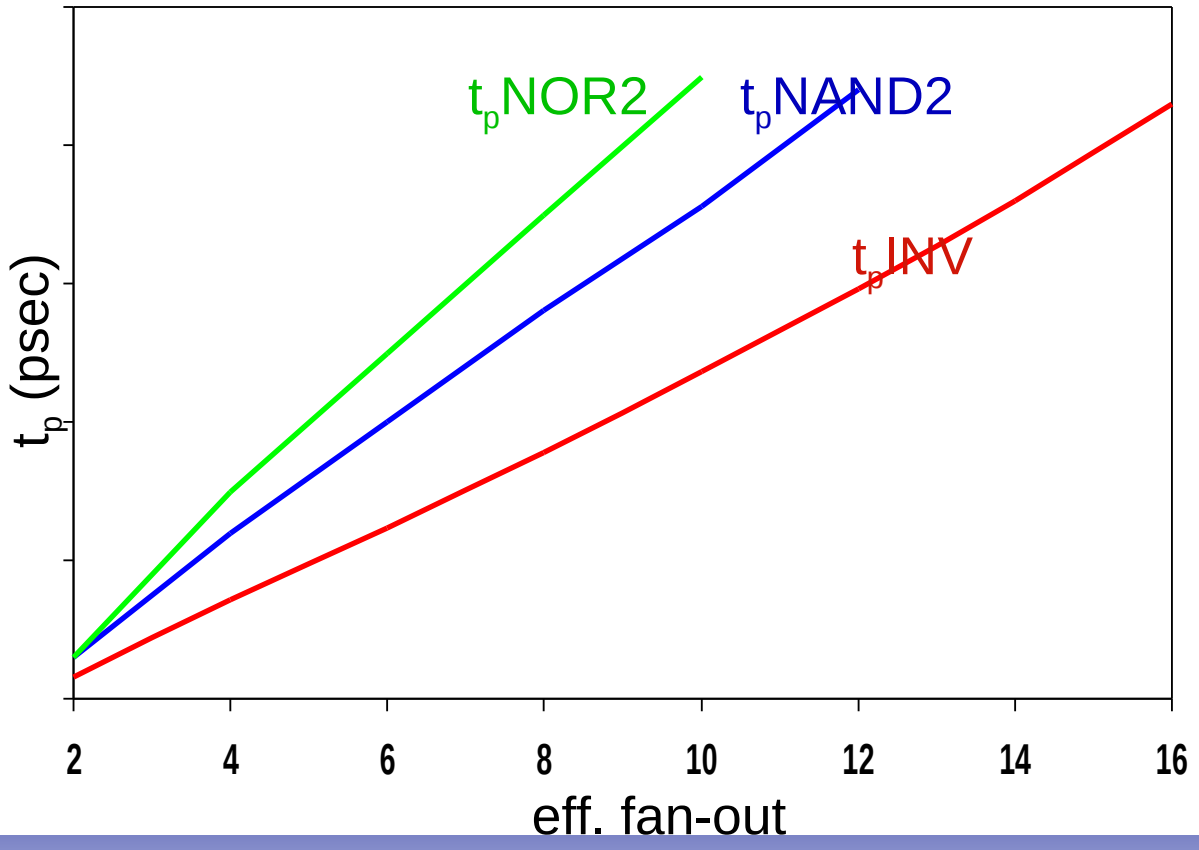
Propagation delay deteriorates rapidly as a function of fan-in – quadratically in the worst case.

t_p as a Function of Fan-In



Gates with a fan-in greater than 4 should be avoided.

t_p as a Function of Fan-Out



All gates have the same drive current.

Slope is a function of "driving strength"

t_p as a Function of Fan-In and Fan-Out

- Fan-in: quadratic due to increasing resistance and capacitance
- Fan-out: each additional fan-out gate adds two gate capacitances to C_L

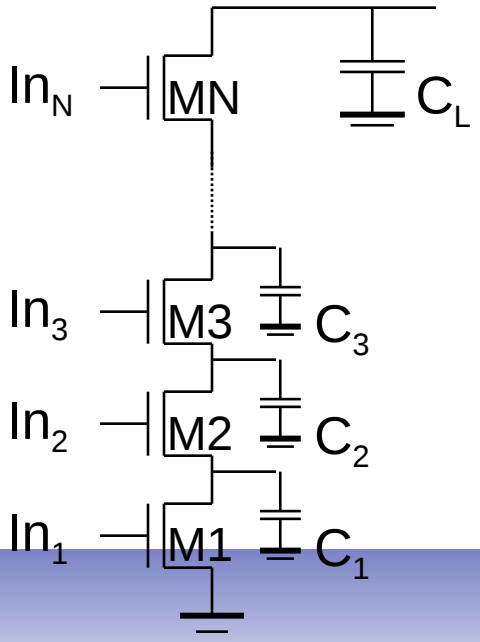
$$t_p = a_1 FI + a_2 FI^2 + a_3 FO$$

Practical Optimization

- The previous arguments regarding tp raise the question – why build nor at all?
  - Criticality is not a path– but a transition so it is usually only on rising or falling (but not both)
  - NOR forms have bad pull-up but good pull down
  - NAND forms have bad pull-down but good pull up
  - Determine the critical transition(s) and design for them– using Elmore or Simulation on the appropriate edge only!
  - Logical Effort presupposes uniform rise and fall times, so good in general, but can be beat
- Static Timing Analyzers nearly always get this wrong!

Fast Complex Gates: Design Technique 1

- Transistor sizing
  - as long as fan-out capacitance dominates
- Progressive sizing



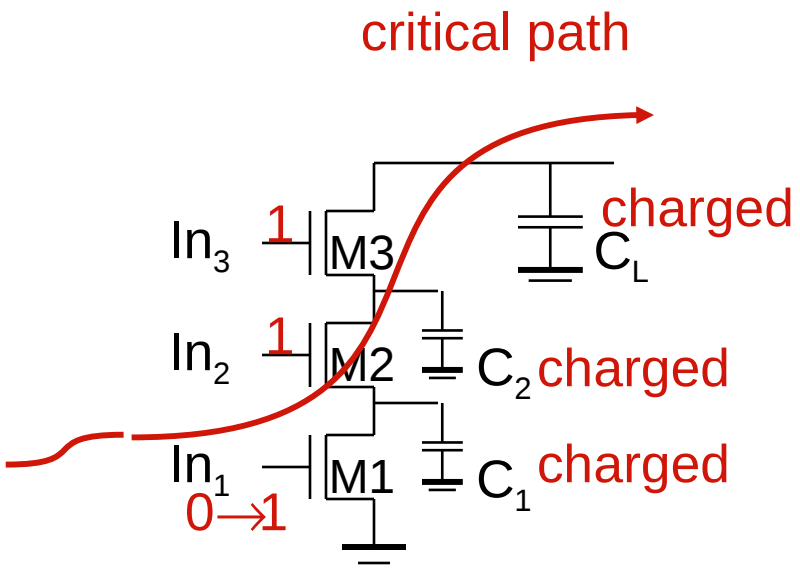
Distributed RC line

M1 > M2 > M3 > … > MN
(the fet closest to the output is the smallest)

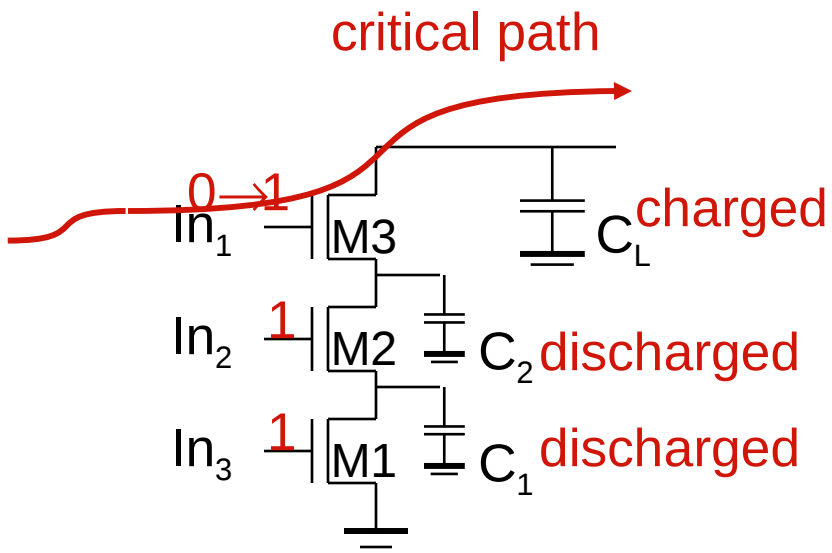
Can reduce delay by more than 20%; decreasing gains as technology shrinks

Fast Complex Gates: Design Technique 2

- Transistor ordering



delay determined by time to discharge C_L, C_1 and C_2

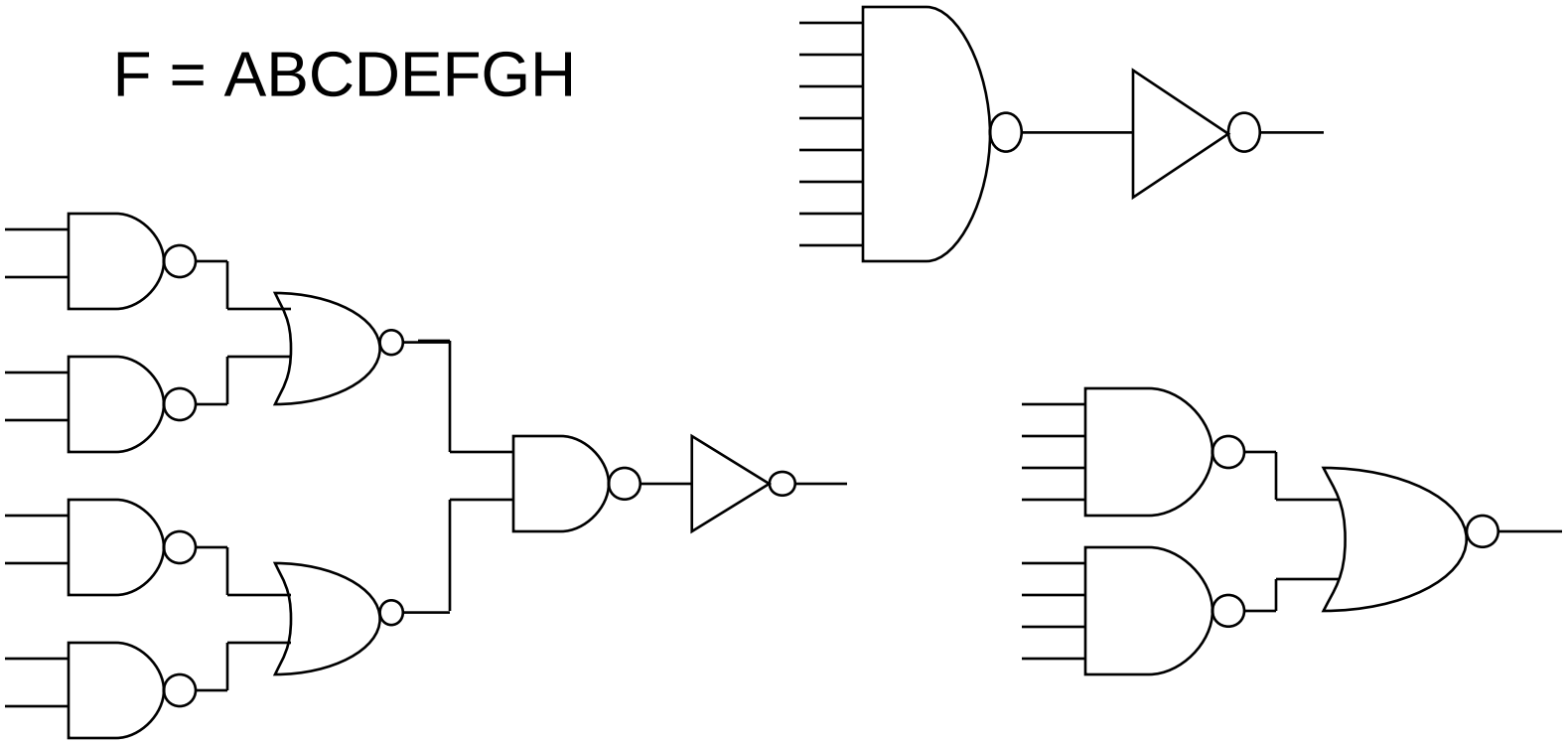


delay determined by time to discharge C_L

Fast Complex Gates: Design Technique 3

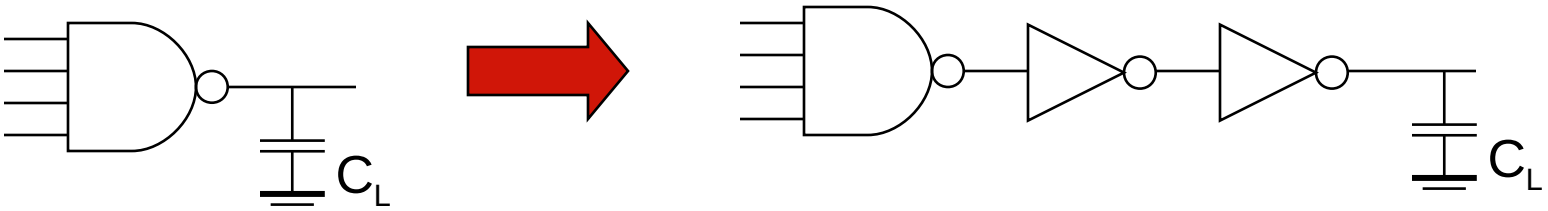
- Alternative logic structures

F = ABCDEFGH



Fast Complex Gates: Design Technique 4

- Isolating fan-in from fan-out using buffer insertion



Fast Complex Gates: Design Technique 5

- Reducing the voltage swing

$$t_{pHL} = 0.69\,(3/4\,(C_L\,V_{DD})/\,I_{DSATn}\,)$$

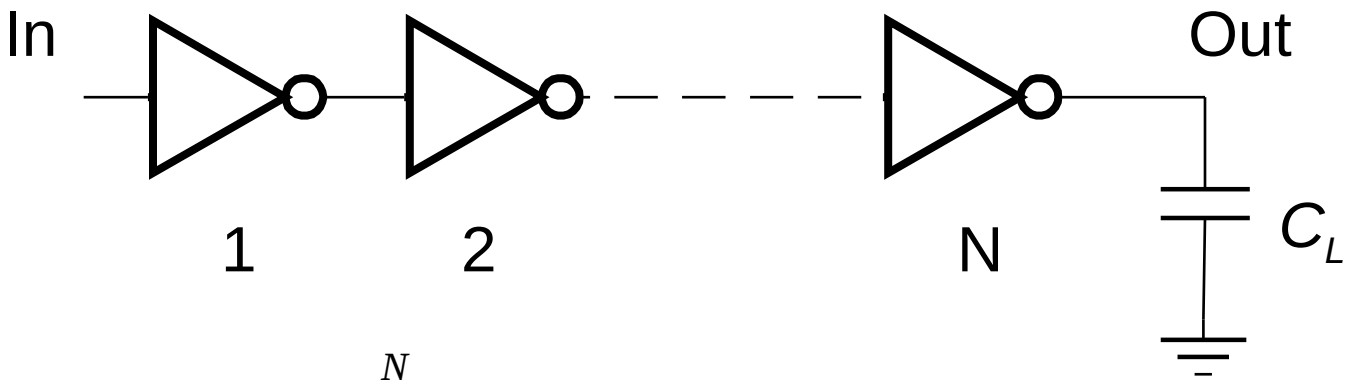
$$= 0.69\,(3/4\,(C_L\,V_{swing})/\,I_{DSATn}\,)$$

  - linear reduction in delay
  - also reduces power consumption
- But the following gate may be much slower!
- Large fan-in/fan-out requires use of “sense amplifiers” to restore the signal (memory)

Sizing Logic Paths for Speed

- Frequently, input capacitance of a logic path is constrained
- Logic also has to drive some capacitance
- Example: ALU load in an Intel's microprocessor is 0.5pF
- How do we size the ALU datapath to achieve maximum speed?
- We have already solved this for the inverter chain – can we generalize it for any type of logic?

Buffer Example



$$Delay = \sum_{i=1}^{N} \left(p_i + g_i \cdot f_i \right)$$

(in units of τ_{inv})

For given N: $C_{i+1}/C_i = C_i/C_{i-1}$

To find N: $C_{i+1}/C_i \sim 4$

How to generalize this to any logic path?

Logical Effort

$$Delay = k \cdot R_{unit} C_{unit} \left(1 + \frac{C_L}{\gamma C_{in}} \right)$$

$$= \tau \left(p + g \cdot f \right)$$

p – intrinsic delay ($3kR_{unit}C_{unit}\gamma$) - gate parameter ≠ f(W)

g – logical effort ($kR_{unit}C_{unit}$) – gate parameter ≠ f(W)

f – effective fanout

Normalize everything to an inverter:

g_{inv} =1, p_{inv} = 1

Divide everything by τ_{inv}

(everything is measured in unit delays τ_{inv})

Assume γ = 1.

Delay in a Logic Gate

Gate delay:

$$d = h + p$$

effort delay — **intrinsic delay**

Effort delay:

$$h = g\,f$$

logical effort — **effective fanout = C_{out}/C_{in}**

Logical effort is a function of topology, independent of sizing

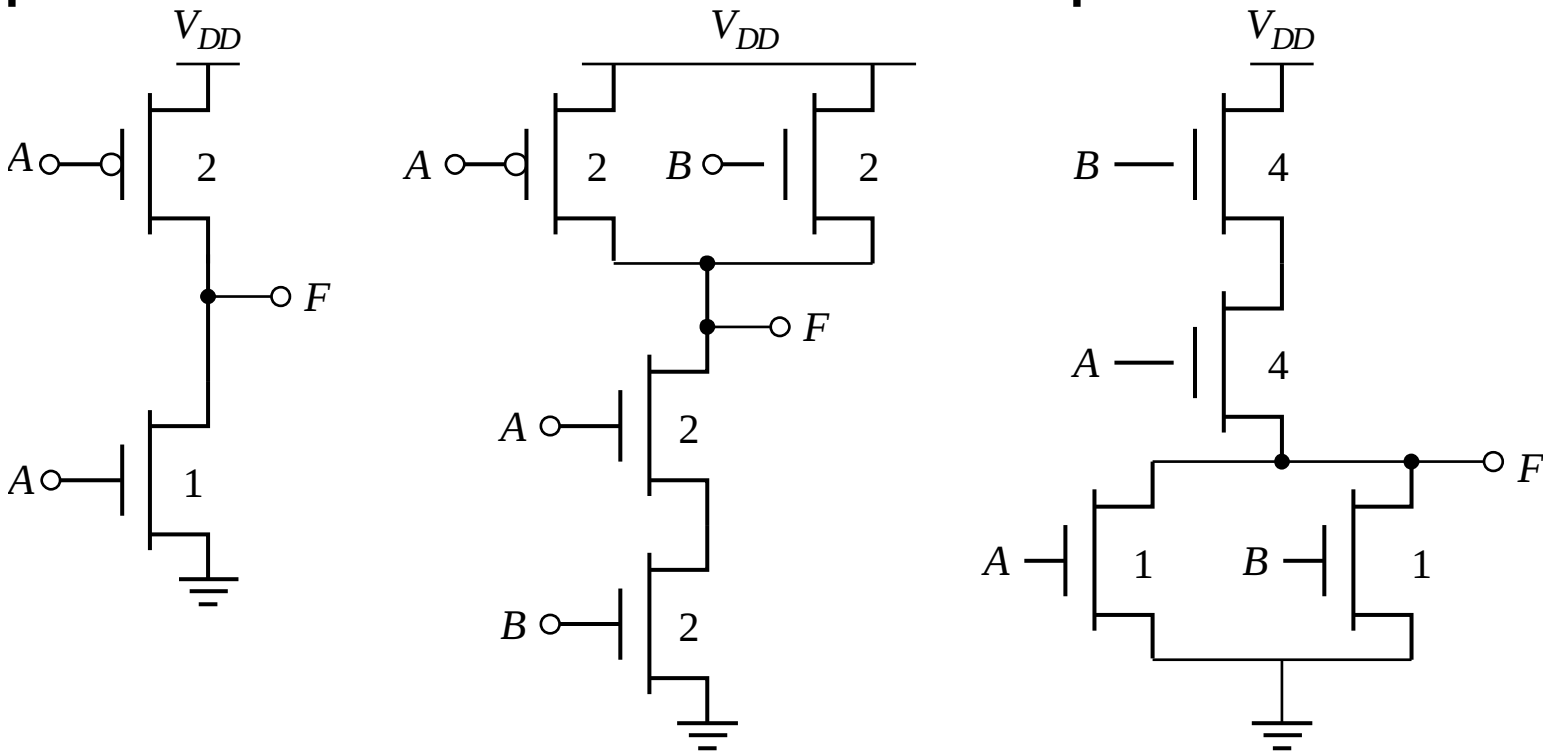
Effective fanout (electrical effort) is a function of load/gate size

Logical Effort

- Inverter has the smallest logical effort and intrinsic delay of all static CMOS gates
- Logical effort of a gate presents the ratio of its input capacitance to the inverter capacitance when sized to deliver the same current
- Logical effort increases with the gate complexity

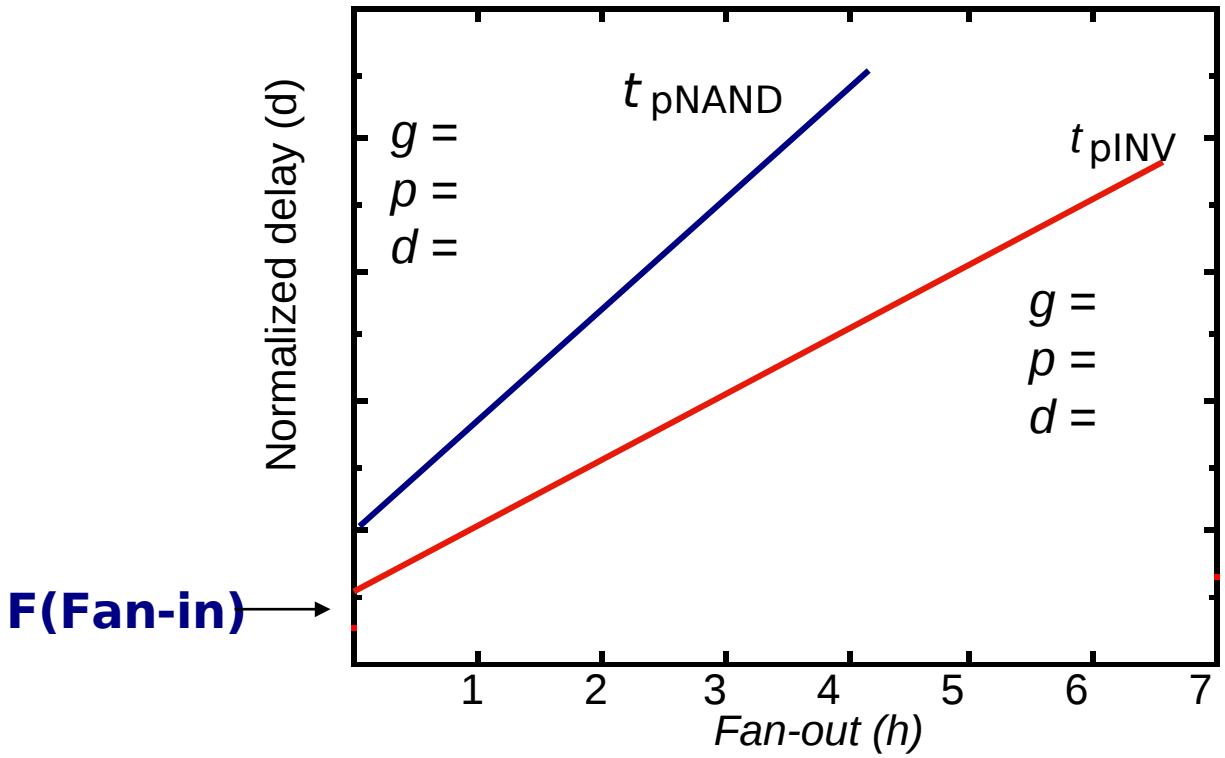
Logical Effort

Logical effort is the ratio of input capacitance of a gate to the input capacitance of an inverter with the same output current

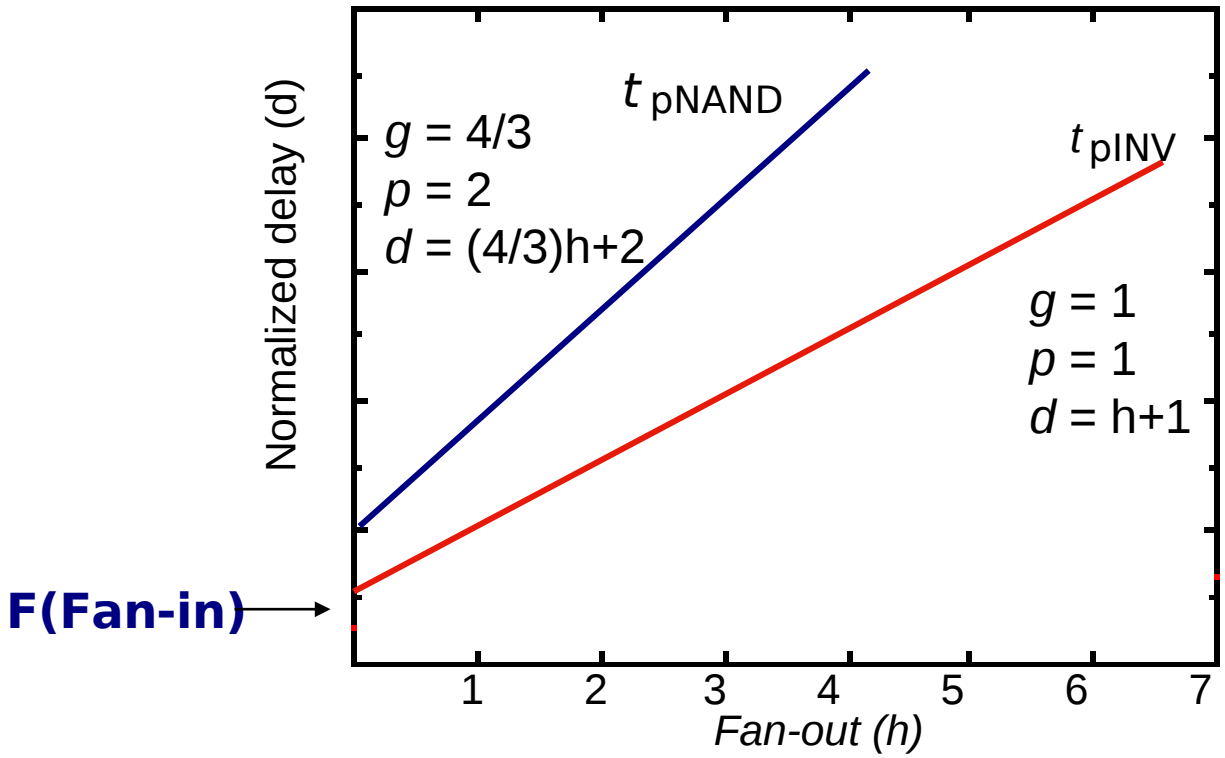


***g* = 1** ***g* = 4/3** ***g* = 5/3**

Logical Effort of Gates



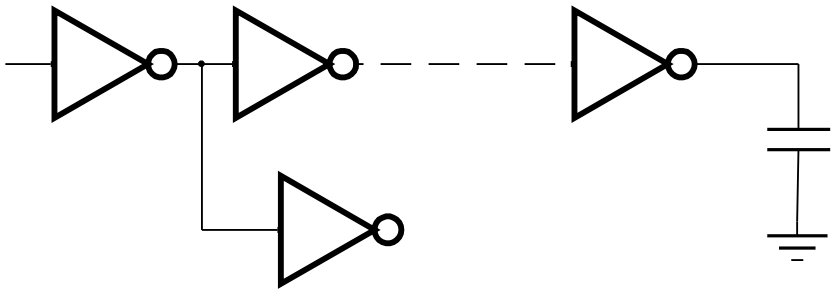
Logical Effort of Gates



Add Branching Effort

Branching effort:

$$b = \frac{C_{on-path} + C_{off-path}}{C_{on-path}}$$



Multistage Networks

$$Delay = \sum_{i=1}^{N} \left(p_i + g_i \cdot f_i \right)$$

Stage effort: $h_i = g_i f_i$

Path electrical effort: $F = C_{out}/C_{in}$

Path logical effort: $G = g_1 g_2 \ldots g_N$

Branching effort: $B = b_1 b_2 \ldots b_N$

Path effort: $H = GFB$

Path delay $D = \Sigma d_i = \Sigma p_i + \Sigma h_i$

Optimum Effort per Stage

When each stage bears the same effort:

$$h^N = H$$

$$h = \sqrt[N]{H}$$

Stage efforts: $g_1 f_1 = g_2 f_2 = \ldots = g_N f_N$

Effective fanout of each stage: $f_i = h / g_i$

Minimum path delay

$$\hat{D} = \sum \left(g_i f_i + p_i \right) = NH^{1/N} + P$$

Optimal Number of Stages

For a given load,
and given input capacitance of the first gate
Find optimal number of stages and optimal sizing

$$D = NH^{1/N} + Np_{inv}$$

$$\frac{\partial D}{\partial N} = -H^{1/N}\ln\left(H^{1/N}\right) + H^{1/N} + p_{inv} = 0$$

Substitute 'best stage effort' $h = H^{1/\hat{N}}$

Logical Effort

Gate Type	Number of Inputs			
	1	2	3	n
Inverter	1			
NAND		4/3	5/3	(n + 2)/3
NOR		5/3	7/3	(2n + 1)/3
Multiplexer		2	2	2
XOR		4	12	

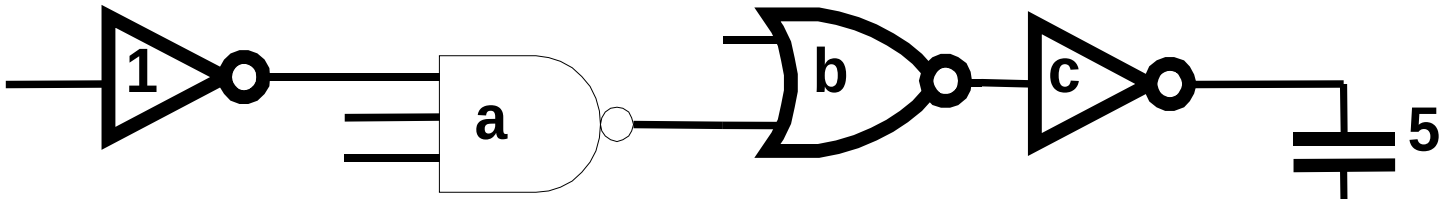
From Sutherland, Sproull

Method of Logical Effort

- Compute the path effort: $F = GBH$
- Find the best number of stages $N \sim \log_4 F$
- Compute the stage effort $f = F^{1/N}$
- Sketch the path with this number of stages
- Work either from either end, find sizes: $C_{in} = C_{out} * g/f$

Reference: Sutherland, Sproull, Harris, “Logical Effort, Morgan-Kaufmann 1999.

Example: Optimize Path



$g = 1$ $f = a$

$g = 5/3$ $f = b/a$

$g = 5/3$ $f = c/b$

$g = 1$ $f = 5/c$

Effective fanout, F =

G =

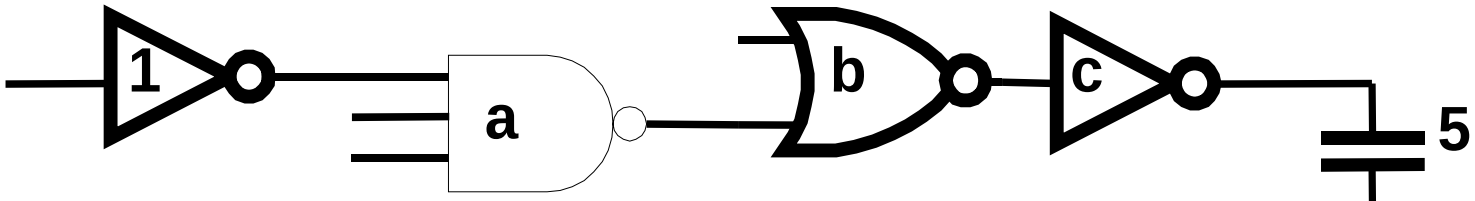
H =

h =

a =

b =

Example: Optimize Path



$g = 1$ $f = a$

$g = 5/3$ $f = b/a$

$g = 5/3$ $f = c/b$

$g = 1$ $f = 5/c$

Effective fanout, $F = 5$

$G = 25/9$

$H = 125/9 = 13.9$

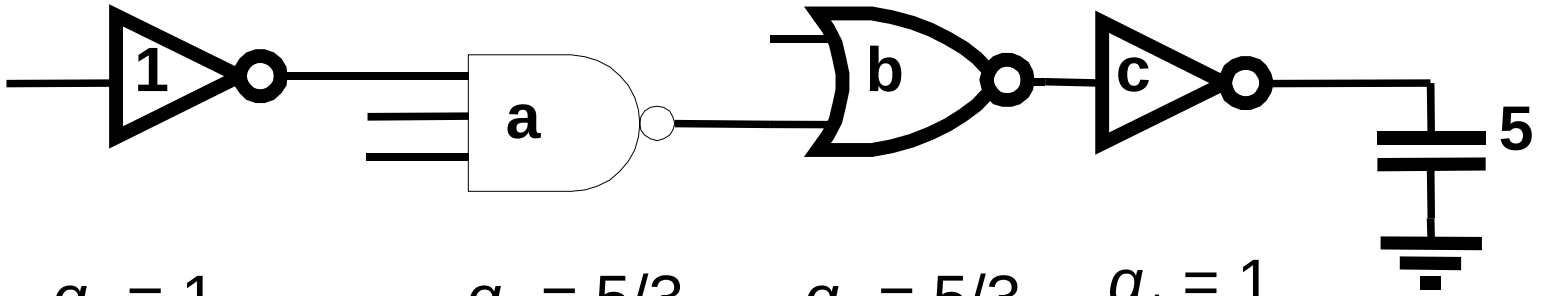
$h = 1.93$

$a = 1.93$

$b = ha/g_2 = 2.23$

$c = hb/g_3 = 5g_4/f = 2.59$

Example: Optimize Path



$g_1 = 1$ $g_2 = 5/3$ $g_3 = 5/3$ $g_4 = 1$

Effective fanout, H = 5

$G = 25/9$

$F = 125/9 = 13.9$

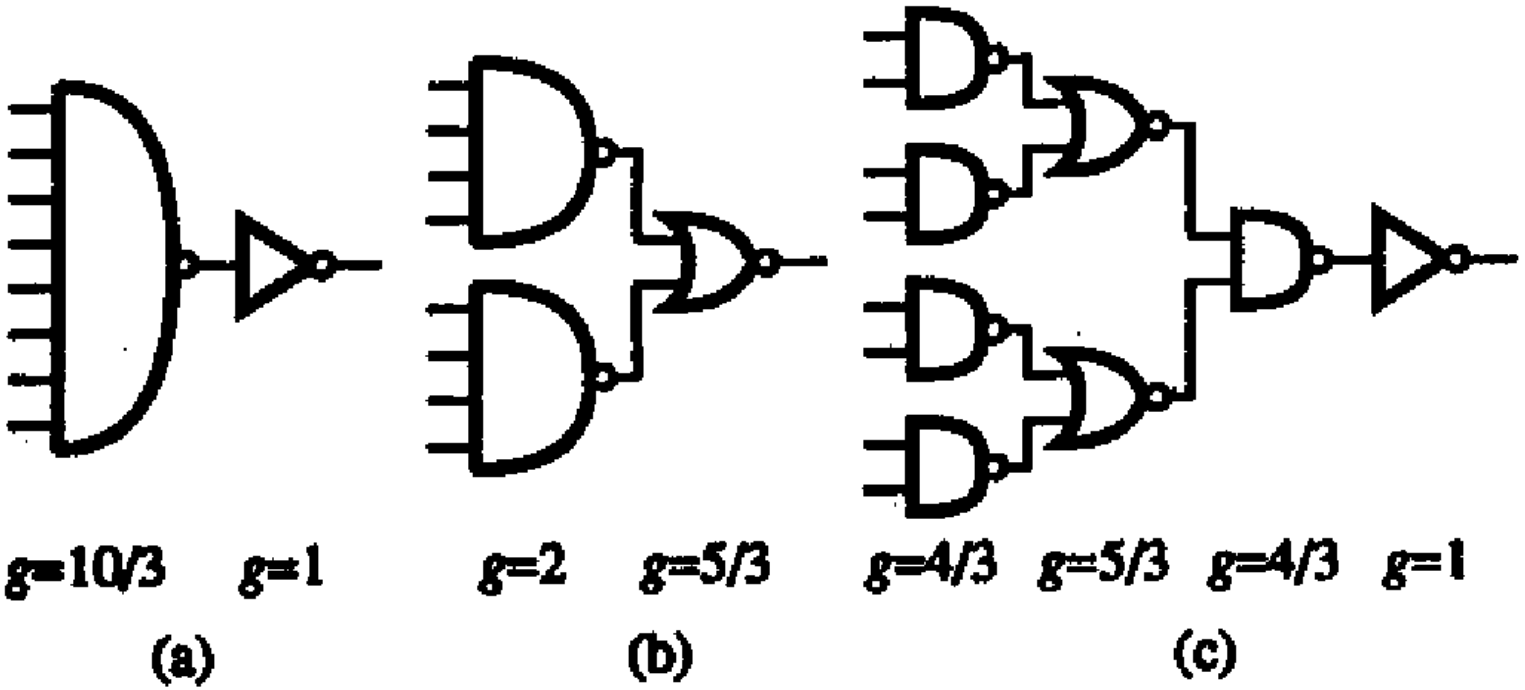
$f = 1.93$

$a = 1.93$

$b = fa/g_2 = 2.23$

$c = fb/g_3 = 5g_4/f = 2.59$

Example – 8-input AND



Summary

Table 4: Key Definitions of Logical Effort

Term	Stage expression	Path expression
Logical effort	g (seeTable 1)	$G = \prod g_i$
Electrical effort	$h = \frac{C_{out}}{C_{in}}$	$H = \frac{C_{out\ (path)}}{C_{in\ (path)}}$
Branching effort	n/a	$B = \prod b_i$
Effort	$f = gh$	$F = GBH$
Effort delay	f	$D_F = \sum f_i$
Number of stages	1	N
Parasitic delay	p (seeTable 2)	$P = \sum p_i$
Delay	$d = f + p$	$D = D_F + P$

Sutherland,
Sproull
Harris

Homework 5

1. Using the Carry cell design from earlier homework, optimally size the carry propagate chain for a 16-bit adder to minimize worst case delay where Cin is driven by a 1u/0.6u inverter and Cout drives a fanout of 4 such loads. (use logic effort, show your work!)
2. For the problems below, use parameters from class for 0.5um and use 2x voltages as applicable. Chap 5: problems: 4, 7, 8, 15
3. Chap 6, problems: 2, 4, 5, 7
4. Design the parity tree: c = a xor b xor c xor d in Complementary Pass Transistor Logic, insert inverters to restore the output swing – Given input drive from an inverter stage, and an inverter every 2 stages of logic, and inverter output restore, estimate the propagation time for devices using the AMI 0.5um model.

New (digital) AMI model (for minimum length only!):

n-channel: V_T=0.77, λ=0.03, V_{sat}=1.56V, k=32μA/V^2

p-channel: V_T=-0.95, λ=0.03 V_{sat}=2.8V, k=-16μA/V^2